

由浅入深，学练结合，轻松掌握C语言编程

Broadview®
www.broadview.com.cn

• 本书特色 •

基础入门→进阶开发→编程实践→项目实战
170个典型实例、436个练习题、2个项目案例

• 超值、大容量DVD •

5小时多媒体教学视频
本书源代码、本书教学PPT
赠送19小时相关学习视频

• 本书技术支持 •

答疑QQ群：21948169
答疑论坛：<http://www.rzchina.net>



由浅入深学 C语言

——基础、进阶与必做430题



24小时视频讲解

◎崔久 蒋欣 等编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
www.phei.com.cn

由浅入深学

由浅入深学 C语言

——基础、进阶与必做430题

◎崔久 蒋欣 等编著

電子工業出版社·

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

目前 C 语言已经成为世界上最流行的高级语言之一,具有简单易学、结构化、可对计算机硬件直接操作、可移植性强等特性,被应用于计算机的各个领域,例如系统软件和应用软件的开发、科学计算等方面。

本书简单易懂,内容丰富,包含大量的实例和习题,由易到难逐步讲解,使读者易于了解和掌握本书讲解的知识。本书由 4 篇组成:第 1 篇是 C 语言基础,讲解了 C 语言的发展历程、特点及 C 语言程序的编译和链接、顺序结构、条件结构、循环结构;第 2 篇是 C 语言技术进阶,包括数组、指针、函数、结构型、共用型、枚举型和用户自定义类型;第 3 篇是 C 语言高级应用,包括算法、预编译命令、文件和图形,以及预处理命令等内容;第 4 篇是 C 语言开发案例,详细讲解了 C 语言项目开发案例,读者在这里可以学习到 C 语言项目的整个开发过程。

全书的习题有关键知识点的提示,以帮助读者学习和理解 C 语言的知识点。本书适合正在学习 C 语言的初中级读者阅读,并可作为 C 语言项目开发人员的参考手册。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

由浅入深学 C 语言:基础、进阶与必做 430 题 / 崔久等编著. —北京:电子工业出版社, 2011.7
(由浅入深学)
ISBN 978-7-121-13333-6

I. ①由… II. ①崔… III. ①C 语言—程序设计—习题集 IV. ①TP312-44

中国版本图书馆 CIP 数据核字(2011)第 067036 号

策划编辑:胡辛征

责任编辑:高洪霞 刘娴庆

印 刷:北京东光印刷厂

装 订:三河市皇庄路通装订厂

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1092 1/16 印张:31 字数:796.8 千字

印 次:2011 年 7 月第 1 次印刷

印 数:4000 册 定价:59.80 元(含 DVD 光盘 1 张)

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

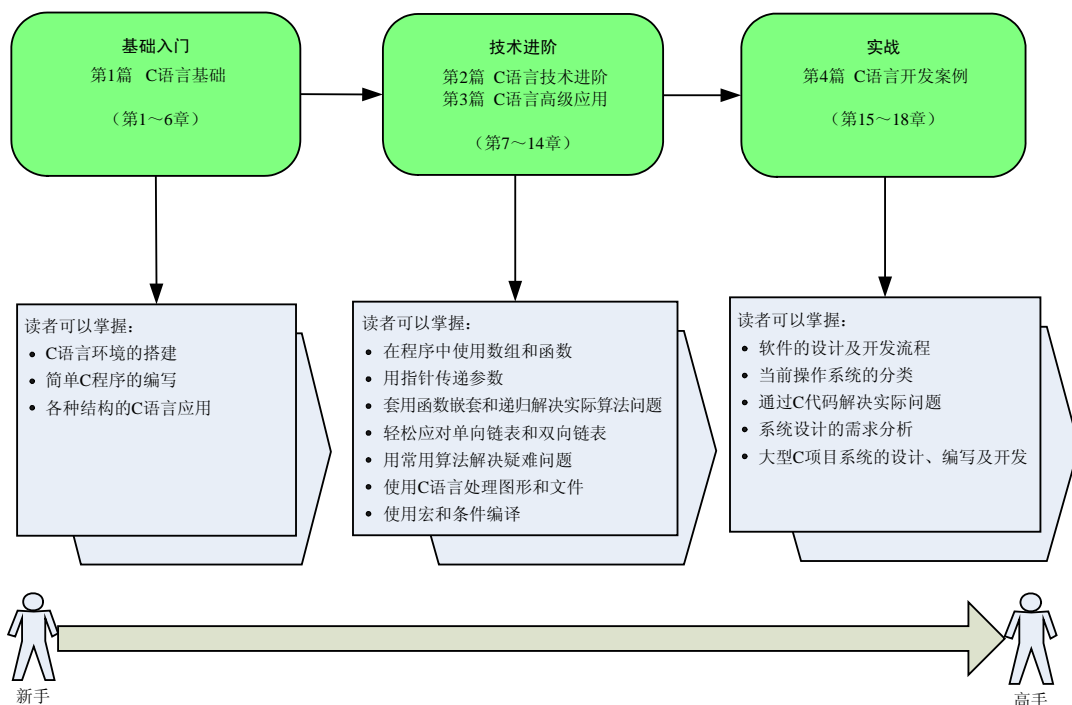
服务热线:(010) 88258888。

前言

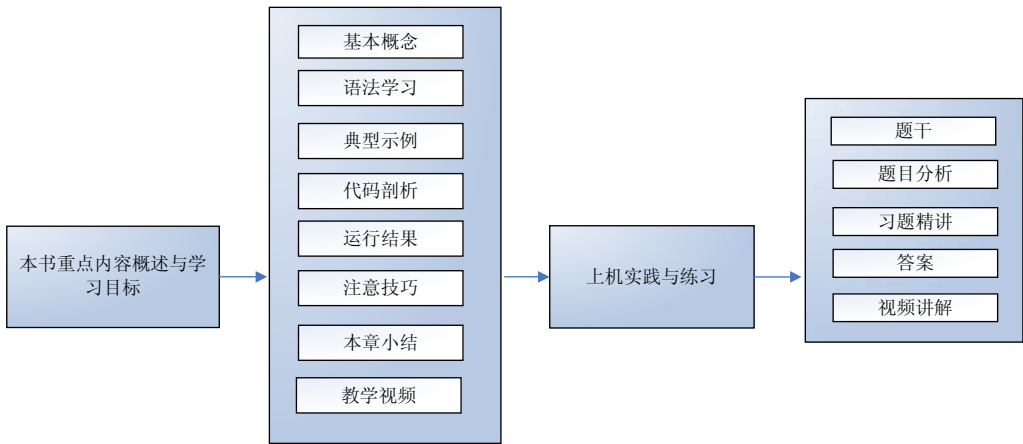
C 语言是一门经久不衰，永远保持着青春与活力的程序设计语言，从产生到现在，它已经成为编程初学者必学的编程语言之一。它具有语言简洁、紧凑，使用方便、灵活，运算符丰富，生成目标代码质量高，程序执行效率高，程序可移植性强（与汇编语言比）的特点，更易于被广大初学者接受。C 语言既有高级语言的特性，又有低级语言的特性（能对计算机硬件直接操作），是一门成功的高级程序设计语言。

为了让对编程有兴趣的人们能尽快入门，作者精心编写了本书，目的是引领普通读者进入这一门槛。从了解 C 语言的基本语法开始，通过上百个 C 语言开发实例和两个 C 语言开发项目讲解 C 语言，让读者可以站在前人的肩膀上学习，得到事半功倍的效果。

本书内容体系与学习梯度



本书内容编写体例



本书的特点

随着 UNIX 系统的广泛使用，C 语言的应用也越来越广泛。现在的 Java、C++、C# 等语言都是由 C 语言发展过来的。

本书的特点主要体现在以下几个方面。

- 简单易学，讲解的有关 C 语言的知识很容易被理解和掌握。
- 实例丰富，每一个重点的知识点至少包含一个以上的实例。
- 知识全面，涉及 C 语言的各个方面。
- 循序渐进，从基础讲解起，到完整项目的开发。
- 习题丰富并带提示，可帮助读者巩固本章学习的知识。

本书的内容安排

本书分为 4 篇，共 18 章，从简单的输出程序开始，进一步介绍 C 语言各个方面的知识，然后通过两个 C 语言项目实例，进一步加深读者对 C 语言的认识。

第 1 篇（第 1 章至第 6 章）C 语言基础。

第 1~3 章讲述了 C 语言的开发环境及其配置、C 语言的一些基础知识、C 语言的发展历程、特点，以及 C 语言程序的编译和链接等内容。从第 4 章开始讲述了顺序结构、条件结构、循环结构的概念及其使用，其中条件结构包括 if 单分支结构、if 多分支结构、switch 结构这三种结构。学完本篇，读者可以编写简单的 C 语言程序。

第 2 篇（第 7 章至第 10 章）C 语言技术进阶。

讲解了数组、指针、函数、结构型、共用型、枚举型和用户自定义类型等内容。学完本章，读者将对 C 语言的数据结构有进一步的了解。

第 3 篇（第 11 章至第 14 章）C 语言高级应用。

讲解了 C 语言中的一些基本算法、预编译命令、文件及图形等内容。学习完本篇后，读者可以深层次地了解 C 语言知识。

第4篇（第15章至第18章）C语言开发案例。

本篇开始介绍了系统开发的一些知识，然后通过一些案例了解C语言的算法。最后第17~18章提供了两个案例：航空订票管理系统和学习管理系统。学习完本篇后，读者可以编译大型的C语言开发项目案例。

光盘说明



为帮助读者直观地学习，本书附赠DVD光盘，内容包含数十小时多媒体视频、电子教案（PPT）、实例源代码、职场面试法宝等内容。

适合阅读本书的读者




- C语言初、中级学员；
- C语言开发设计人员；
- 高等学校学习C语言的学生；
- 对C语言有简单了解，但不全面的人员；
- 从其他语言转过来学习C语言的开发人员。

目 录




第 1 篇 C 语言基础


第 1 章 第一个 C 语言程序 ( 教学视频: 14 分钟)	1
1.1 搭建开发环境	1
1.1.1 对硬件系统的要求	1
1.1.2 对软件环境的要求	2
1.1.3 C 语言开发工具简介	2
1.2 第一个 C 语言程序	4
1.2.1 学习 C 语言的好工具 Visual C++	4
1.2.2 创建及运行第一个程序	5
1.3 良好的代码规范	8
1.3.1 规范命名	8
1.3.2 美观对称	9
1.3.3 合理注释	10
1.4 小结	11
1.5 习题	12
第 2 章 C 语言基础 ( 教学视频: 16 分钟)	15
2.1 程序语言基础知识	15
2.2 C 语言简介	15
2.2.1 C 语言发展史	16
2.2.2 C 语言特点	16
2.2.3 C 语言结构	17
2.3 C 程序举例及其构成	17
2.4 C 程序的编译和执行	21
2.4.1 编译程序	21
2.4.2 解释程序	22
2.4.3 分块编译	23
2.4.4 函数和连接	24
2.4.5 运行程序	25
2.5 算法设计与分析	25
2.5.1 算法简介	25

2.5.2 算法复杂性	26
2.6 小结	26
2.7 习题	26
第3章 变量和数据类型 ( 教学视频: 16 分钟)	31
3.1 常量及符号常量	31
3.1.1 常量	31
3.1.2 符号常量	32
3.2 变量	33
3.2.1 变量的概念及定义	33
3.2.2 变量地址	34
3.2.3 变量初始化	35
3.3 C 语言的基本数据类型	35
3.3.1 整型常量	36
3.3.2 整型变量	37
3.3.3 浮点型	39
3.3.4 字符型	41
3.4 数据机内存储形式	43
3.4.1 整型数据机内存储形式	43
3.4.2 浮点型数据机内存储形式	43
3.4.3 字符型数据机内存储形式	43
3.5 局部变量	44
3.6 全局变量	44
3.7 形式参数	45
3.8 赋值及类型转换	46
3.9 运算符及其表达式	46
3.9.1 算术运算符及其表达式	47
3.9.2 加 1 和减 1 运算符	47
3.9.3 关系运算符及其表达式	49
3.9.4 逻辑运算符及其表达式	50
3.9.5 三目运算符	50
3.9.6 位运算符	51
3.9.7 sizeof 运算符	53
3.9.8 逗号运算符	54
3.10 小结	55
3.11 习题	55

第 4 章 顺序结构程序设计 ( 教学视频: 18 分钟)	62
4.1 顺序结构程序设计初探	62
4.1.1 顺序结构流程图和 N-S 流程图	62
4.1.2 简单的顺序结构程序	62
4.1.3 了解 C 语言的格式输入、输出函数	64
4.2 详解格式输入、输出函数	65
4.2.1 调用 scanf()函数实现格式化输入	65
4.2.2 调用 printf()函数实现格式化输出	67
4.2.3 putchar()函数	69
4.2.4 getchar()函数	70
4.3 本章技术点范例应用	72
4.4 本章综合练习	72
4.5 小结	73
4.6 习题	74
第 5 章 条件结构程序设计 ( 教学视频: 15 分钟)	83
5.1 条件结构简介	83
5.1.1 if 单分支形式	83
5.1.2 if-else 双分支形式	85
5.2 多重 if	86
5.3 嵌套 if	88
5.4 switch 结构	89
5.5 实战项目	91
5.6 小结	94
5.7 习题	95
第 6 章 循环结构程序设计 ( 教学视频: 19 分钟)	109
6.1 for 循环	109
6.1.1 for 循环	109
6.1.2 for 循环结构应用	111
6.2 while 循环	114
6.3 do-while 循环	116
6.4 三种循环结构的区别	118
6.5 嵌套循环	120
6.6 循环结构强化实例	123
6.7 小结	126
6.8 习题	126



第2篇 C语言技术进阶



第7章 数组 ( 教学视频: 17 分钟)	140
7.1 数组简介	140
7.2 为何需要数组	141
7.3 一维数组	141
7.3.1 一维数组的声明和初始化	141
7.3.2 一维数组的引用	143
7.4 二维数组	146
7.4.1 二维数组的声明和初始化	147
7.4.2 二维数组应用举例	148
7.5 字符数组	151
7.5.1 字符串与字符数组	151
7.5.2 字符串输入、输出函数	152
7.5.3 字符串函数	154
7.6 数组实战项目	155
7.7 小结	158
7.8 习题	158
第8章 指针 ( 教学视频: 18 分钟)	169
8.1 指针简介	169
8.2 指针的定义及应用	170
8.2.1 指针的定义	170
8.2.2 指针的引用	170
8.2.3 指针变量作为函数的参数	173
8.3 指针与数组	174
8.3.1 指针和一维数组	174
8.3.2 指针和二维数组	177
8.4 指针和字符串	179
8.5 函数的指针	181
8.6 指向指针的指针	183
8.7 指针应用举例	184
8.8 小结	187
8.9 习题	187
第9章 函数 ( 教学视频: 25 分钟)	200
9.1 函数定义和调用	200

9.1.1	定义函数	200
9.1.2	调用函数	202
9.1.3	函数的返回值	203
9.2	变量的生存期和作用域	205
9.2.1	函数内部变量	205
9.2.2	函数外部变量	206
9.3	函数的实参和形参	209
9.3.1	传值方式	209
9.3.2	传址方式	210
9.4	函数的嵌套和递归	212
9.4.1	函数的嵌套	213
9.4.2	函数的递归	214
9.5	函数应用举例	215
9.6	小结	217
9.7	习题	217
第 10 章	结构型、共用型、枚举型及用户自定义型数据	
	( 教学视频: 17 分钟)	229
10.1	结构体类型	229
10.1.1	结构体类型简介	229
10.1.2	结构体类型定义	230
10.1.3	结构体类型引用	231
10.1.4	结构体变量初始化	233
10.2	结构体数组	234
10.2.1	结构体数组定义	234
10.2.2	结构体数组引用	235
10.2.3	结构体数组初始化	236
10.3	结构指针	237
10.3.1	结构体指针概念及其定义	237
10.3.2	结构体数组指针	238
10.3.3	结构体指针应用	240
10.4	结构与函数参数	242
10.4.1	结构变量作为函数参数	242
10.4.2	结构体地址作为函数参数	244
10.4.3	结构体数组作为函数参数	245
10.5	共用体	247
10.5.1	共用体概念及其定义	247

10.5.2	共同体变量应用	247
10.5.3	共同体与结构体的嵌套	249
10.6	枚举型	251
10.7	用户自定义类型	253
10.8	链表	254
10.8.1	单向链表	255
10.8.2	创建及输出链表	256
10.8.3	双向链表	258
10.8.4	链表中插入结点和删除结点	260
10.9	小结	262
10.10	习题	262

第 3 篇 C 语言高级应用



第 11 章	程序的灵魂——算法 ( 教学视频: 17 分钟)	275
11.1	了解算法的必要性	275
11.2	求最大值算法	276
11.3	求最小值算法	277
11.4	排序算法	278
11.4.1	直接插入排序	278
11.4.2	折半插入排序	280
11.4.3	希尔排序	281
11.4.4	冒泡排序	283
11.4.5	选择排序	285
11.4.6	归并排序	286
11.5	查找算法	289
11.5.1	顺序查找	289
11.5.2	折半查找	290
11.5.3	分块查找	291
11.6	小结	292
11.7	习题	292
第 12 章	文件 ( 教学视频: 17 分钟)	303
12.1	文件简介	303
12.1.1	缓冲文件	303
12.1.2	非缓冲文件	304
12.1.3	文件指针和位置指针	304

12.2	与文件有关的库函数	304
12.2.1	文件的打开和关闭函数	304
12.2.2	文件的读写函数	306
12.3	文件定位函数	315
12.3.1	feof()函数	315
12.3.2	rewind()函数	316
12.3.3	fseek()函数和文件随机存取	317
12.3.4	ftell()函数	319
12.4	出错检测函数	320
12.4.1	ferror()函数	320
12.4.2	clearerr()函数	321
12.5	程序应用举例	321
12.6	小结	324
12.7	习题	324
第 13 章	图形处理基础知识 ( 教学视频: 13 分钟)	334
13.1	C 语言图形基本概念	334
13.2	基本图形函数	334
13.2.1	图形初始化	335
13.2.2	关闭图形函数	336
13.2.3	设置外观函数	336
13.2.4	清除窗口函数	337
13.2.5	清屏函数	337
13.2.6	绘图函数	337
13.3	图形应用范例	341
13.4	小结	343
13.5	习题	343
第 14 章	预处理宏命令 ( 教学视频: 17 分钟)	345
14.1	宏	345
14.1.1	不带参数的宏	345
14.1.2	带参数的宏	348
14.2	文件包含	350
14.3	条件编译	352
14.4	不同存储类型的变量	355
14.4.1	自动类型变量	355
14.4.2	静态变量	355

14.4.3	寄存变量	357
14.4.4	外部变量	359
14.5	程序应用举例	360
14.6	小结	362
14.7	习题	362

第 4 篇 C 语言开发案例

第 15 章	软件设计基础 ( 教学视频: 9 分钟)	371
15.1	程序设计语言基础	371
15.2	操作系统基础知识	373
15.2.1	操作系统分类	374
15.2.2	DOS 简介	375
15.2.3	Windows Server 2000 与 Windows Server 2003	377
15.2.4	Linux	377
15.3	数据库基础知识	378
15.3.1	时下流行的数据管理系统简介	378
15.3.2	SQL 语言简介	380
15.4	软件工程	381
15.4.1	软件概念	381
15.4.2	软件分类	381
15.4.3	软件开发流程	382
15.4.4	软件开发模型	383
15.4.5	软件需求分析	384
15.5	小结	384
15.6	习题	385
第 16 章	C 语言程序综合应用 ( 教学视频: 26 分钟)	386
16.1	八皇后问题	386
16.2	汉洛塔问题	391
16.3	循环赛问题	392
16.4	猴子选大王	394
16.5	三个数的最小公倍数问题	395
16.6	背包问题	398
16.7	马遍历问题	401
16.8	流水线作业问题	403
16.9	迷宫问题	405

16.10	关键路径	407
16.11	小结	410
16.12	习题	410
第 17 章	C 语言开发项目：航空订票管理系统 ( 教学视频：12 分钟)	436
17.1	航空订票管理系统简介	436
17.2	航空订票系统需求分析	436
17.3	航空订票系统可行性分析	437
17.4	航空订票系统总体设计	437
17.4.1	输入、输出航班信息	438
17.4.2	订票功能	438
17.4.3	退票功能	438
17.4.4	查询航班信息	439
17.4.5	删除航班信息	439
17.5	航空订票系统程序设计	440
17.6	小结	449
第 18 章	C 语言开发项目：学生管理系统 ( 教学视频：14 分钟)	450
18.1	学生管理系统需求分析	450
18.2	学生管理系统总体设计	450
18.2.1	添加学生资料	451
18.2.2	查询学生资料	451
18.2.3	排序学生资料	452
18.2.4	删除学生资料	453
18.2.5	修改学生资料	454
18.3	学生管理系统程序实现	455
18.4	小结	470
附录	Visual C++操作技巧小代码	471

第 1 篇 C 语言基础

第 1 章 第一个 C 语言程序

在日常的生活、工作中，语言是人们交流的工具。C 语言（Combined Language）也一样，它是人们用来和计算机交流的语言。C 是一种高级语言，其语法接近于人类语言的习惯。在 C 语言的运行过程中，编译系统经过编译将其转换成机器语言，再在机器中执行从而达到预期的效果。C 语言是现在世界上应用最广泛的几种语言之一，可以用来编写应用软件和系统软件，许多著名的系统软件，如 DBASE IV 和 UNIX 系统都是用 C 语言编写的。

本章主要涉及的知识点有：

- 熟悉 C 语言的语法；
- C 语言的开发环境；
- 集成开发环境（IDE）；
- 代码的规范；
- Visual C++ 工具；
- 程序的注释；
- C 语言关键字。



1.1 搭建开发环境

一个 C 语言源程序要将其变成可执行文件（*.exe），并使其在机器中运行，首先必须配置 C 语言运行的一些基本环境。在本节中将会重点讲述这些问题。

1.1.1 对硬件系统的要求

C 语言对硬件的要求不高，一般的计算机都能满足，但并不是所有的 MCU（微控制单元）都可以用 C 语言来开发，其必须具有一定的硬件条件，如下所示。

- 完整的指令系统；
- 有分别为运算和指针服务的 16 位寄存器；
- 堆栈指针和堆栈结构指针；
- 有连续的存取地址空间。



说明：C 语言适合多种操作系统，如 Windows XP、DOS、UNIX 等，也适用于多种机型。



1.1.2 对软件环境的要求

常用的 C 语言 IDE（集成开发环境）有 Microsoft Visual C++、Dev-C++、Code::Blocks、Borland C++、WaTCom C++、Borland C++ Builder、GNU DJGPP C++、Lccwin32 C Compiler 3.1、High C、Turbo C、C-Free 和 Win-TC 等。

一般只要安装了以上任意一个软件，在操作系统中就可以进行编译及运行用 C 语言编写的程序了。

1.1.3 C 语言开发工具简介

本节将分别介绍几个比较常用的 C 语言开发工具。

1. Visual C++

Visual C++（简称 VC）是一个功能强大的可视化集成开发工具。自从 Microsoft 公司 1993 年推出 Visual C++ 1.0 后，随着版本的不断更新，Visual C++ 已经成为程序员的首选开发工具。最新的 Visual C++ 6.0 版本在编译器、MFC 类库、编辑器及联机帮助系统等方面，都比以前的版本做了很大的改进。

Visual C++ 一般可以分为三个版本：学习版、专业版和企业版，不同的版本适合于不同类型的程序开发。

集成开发环境（IDE）是一个将编译器、程序编辑器、调试工具，以及一些其他建立应用程序的工具集成在一起的，用于开发应用程序的软件系统。Visual C++ 中包含的 Developer Studio 就是一个集成的开发环境，其中包含了各种开发工具和编译器。程序员可单独在该环境对应用程序进行编辑、编译、调试及运行。Developer Studio 中除了程序编辑器、资源编辑器、编译器、调试器之外，还包含各种工具和向导，例如 AppWizard 和 ClassWizard 及 MFC 类库，可以帮助程序员快速正确地开发出应用程序。

Visual C++ 运行界面如图 1-1 所示。

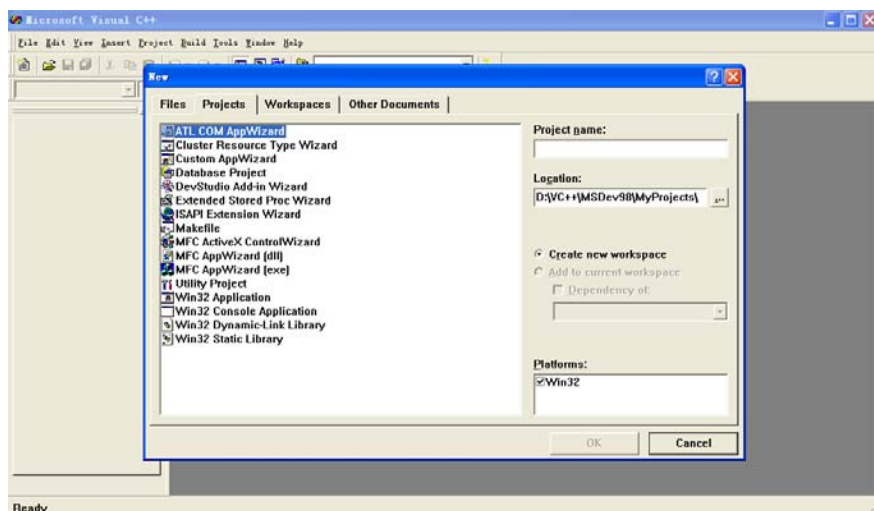


图 1-1 Visual C++ 工作界面



2. Turbo C

Turbo C 是美国 Borland 公司 1987 年开发出来的产品，其中使用了一系列的下拉式菜单，将文本编辑、程序编译、链接及程序运行实现了一体化，在一定程度上方便了程序的开发。

Turbo C 是一个快捷、方便、高效的运行平台，不用单独地编译、链接和运行一个程序，在一个简单屏幕中就可以实现这些功能。但在 Turbo C 中不能使用鼠标，这给编程者带来了一定的困扰。

Turbo C 工作界面如图 1-2 所示。

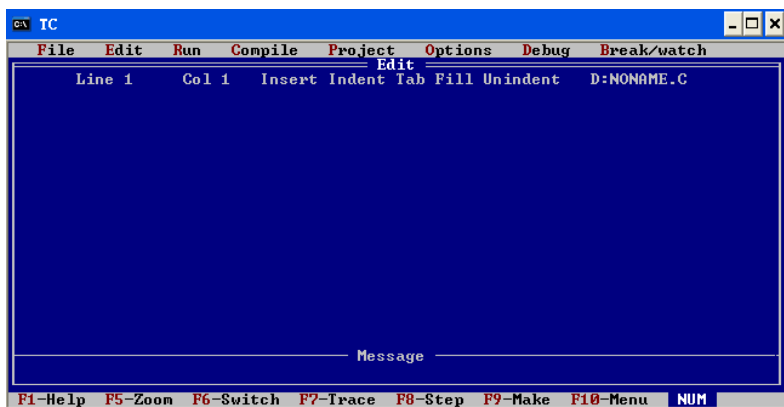


图 1-2 Turbo C 工作界面

3. C-Free

C-Free 是一款集成开发环境软件，适合初学者使用。利用该软件，可以轻松地编辑、编译、链接、运行、调试 C 语言程序。这款软件对于 C/C++ 的学习者非常容易使用，是迅速提高 C 语言水平的好帮手。

C-Free 工作界面如图 1-3 所示。

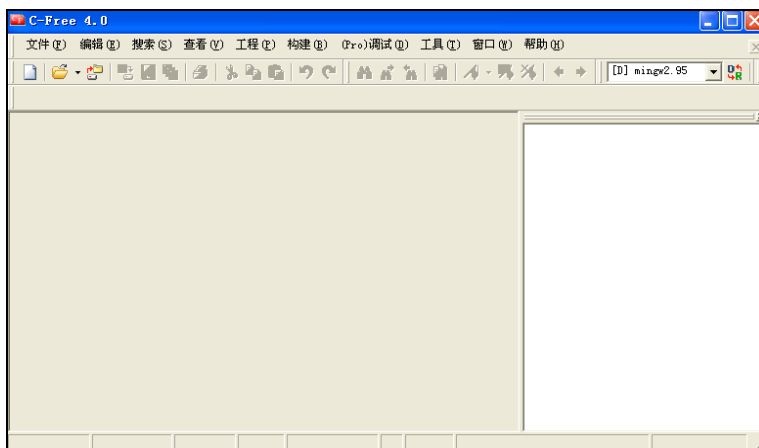


图 1-3 C-Free 工作界面

4. Win-TC

Win-TC，是一个 TC2 Windows 平台开发工具。Win-TC 提供了 Windows 平台的开发界面，



并且有很多的辅助工具，可使用户的编程更加轻松。Win-TC 可在 Windows 的多种操作系统上正常运行，其具有以下功能：

- (1) 支持鼠标操作、程序的复制和粘贴、中文输入输出等功能，使用起来很方便。
- (2) 支持字体大小及颜色的改变。
- (3) 自动设置默认工作目录，一般不用管，当然也可以另外设置。
- (4) 中文界面，提示错误也是中文显示，不用担心看不懂英文的问题。
- (5) 支持结果中显示中文。

Win-TC 工作界面如图 1-4 所示。

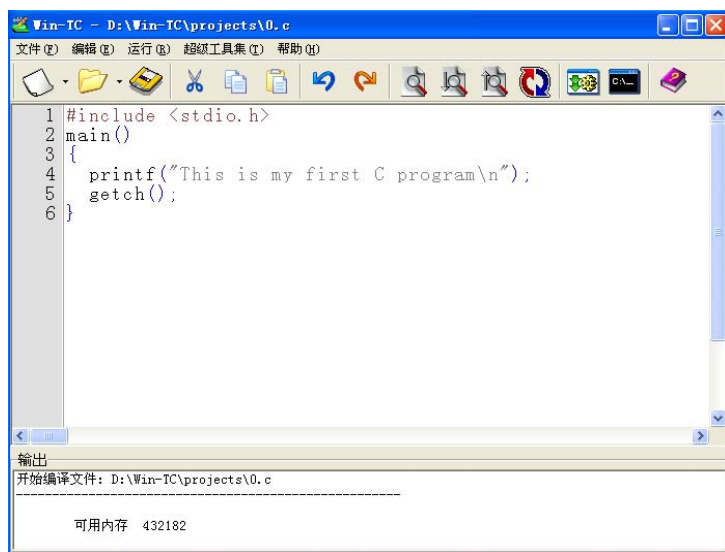


图 1-4 Win-TC 工作界面



1.2 第一个 C 语言程序

在 Visual C++ 中可以对程序进行新建、编辑、编译、链接、运行和调试等操作，在本节中将会讲述如何在 Visual C++ 中进行这些操作，使读者能够初步掌握 C 语言的基础知识，熟悉 Visual C++ 的工作界面。

1.2.1 学习 C 语言的好工具 Visual C++

Visual C++ 作为一个 C 语言编程软件，是一个很好使用并且功能强大的工具。它是目前国内最流行的编译软件之一。

Visual C++ 是一款面向对象的可视化基础编译软件，它包含了丰富的 MFC 类库，其中定义了大量的库函数和类。在进行 Windows 程序的编辑过程中，程序员可以直接调用它来简化程序，使编程量大大减少。Visual C++ 与系统联系非常紧密，使用方法也很灵活，但其开发效率不是很高，这也是 Visual C++ 的一个缺陷。

Visual C++ 主要适用于以下 4 个方面：



- 系统、驱动程序的开发;
- 游戏开发;
- 单片机开发;
- 多线程、网络通信和数据库等方面的应用。

1.2.2 创建及运行第一个程序

用 Visual C++ 可以建立一个工程,也可以建立一个单独的源程序文件。对于简单的 C 语言程序,一般创建一个文件就足够了,下面介绍下 Visual C++ 的使用方法。文件创建过程如下:

- ① 打开 Visual C++ 编译软件。
- ② 选择 Visual C++ 菜单栏“File”|“New”选项,弹出界面如图 1-5 所示。

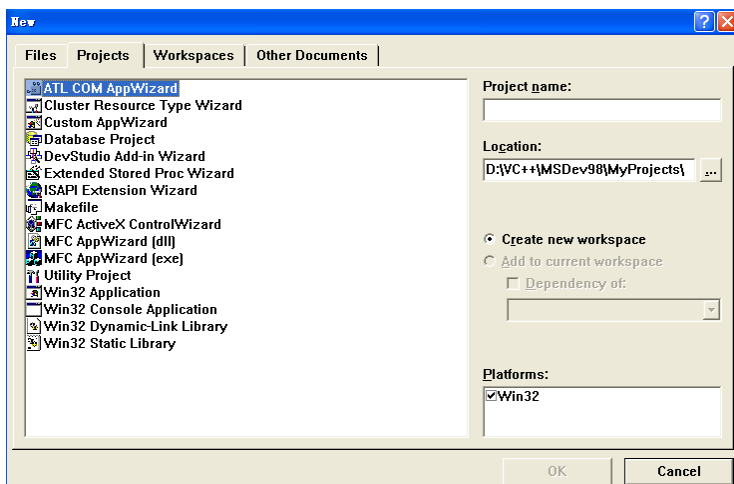


图 1-5 Visual C++ 界面

- ③ 选择“Files”选项卡,在其中选中“C++ Source File”选项,输入文件名 my,如图 1-6 所示。

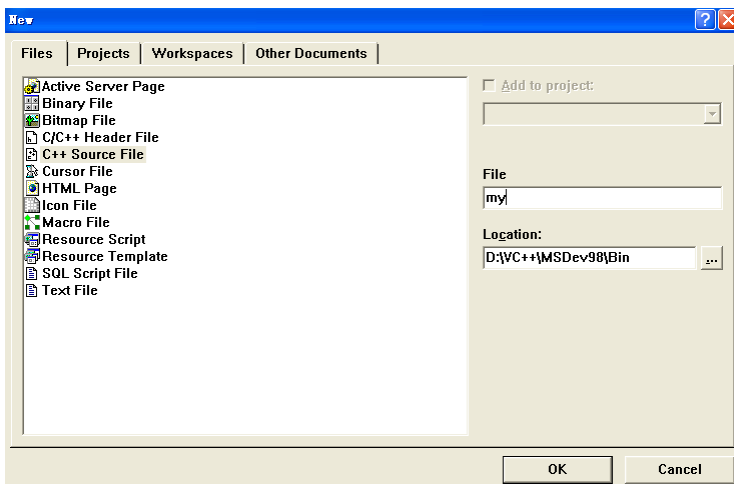


图 1-6 Files 选项



由浅入深学 C 语言——基础、进阶与必做 430 题

- ④ 单击“OK”按钮，在 Visual C++ 中输入以下程序，如图 1-7 所示。

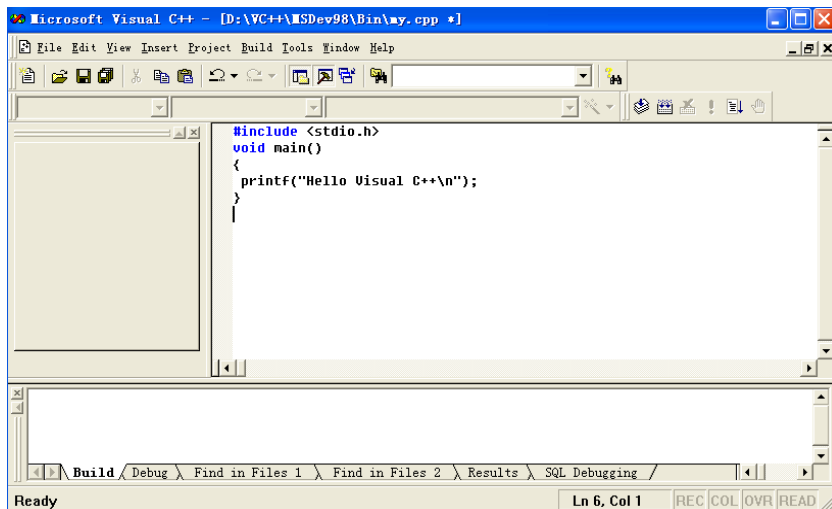


图 1-7 在 Visual C++ 中输入程序

```
#include <stdio.h>
void main()
{
    printf("Hello Visual C++\n");
}
```

- ⑤ 程序编译完后，可以对其进行编译。选择菜单栏“Build”|“Compile my.c”选项，或按下“Ctrl+F7”快捷键可实现对程序的编译，编译后的界面如图 1-8 所示。

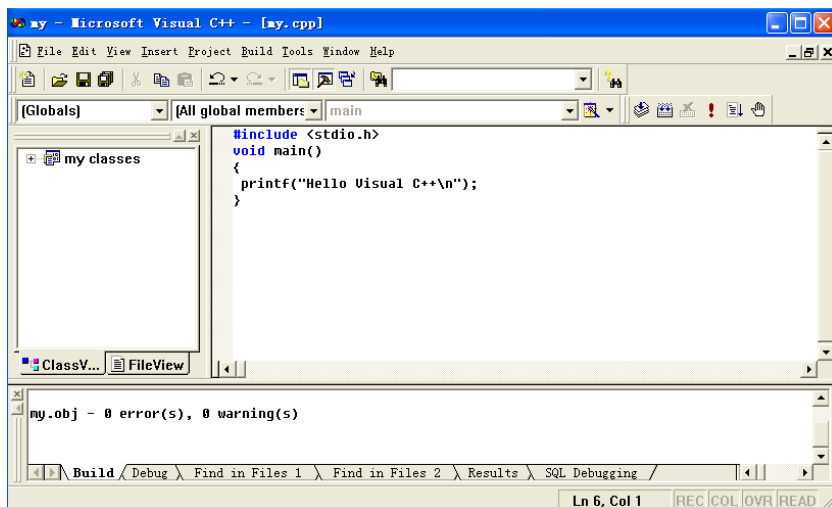


图 1-8 Visual C++ 编译后的界面



提示：若程序无误，则在下方会弹出 0 个 error 和 0 个 warning，表示程序编译没有任何问题，即没有任何错误和警告。编译以后，就可以对程序进行链接了。

- ⑥ 单击菜单栏“Build”|“Build my.exe”选项，或按下“F7”快捷键可链接该程序，其

界面如图 1-9 所示。

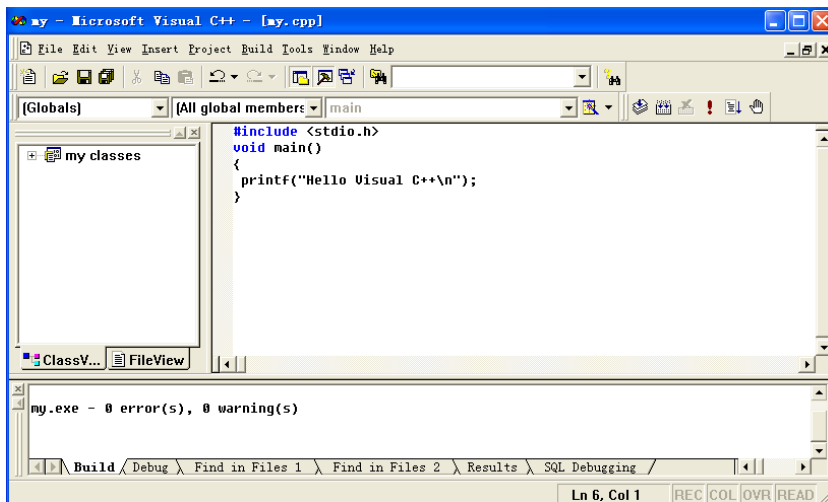


图 1-9 Visual C++链接后界面

⑦ 单击菜单栏“Build”中的“Execute my.exe”，或“Ctrl+F5”快捷键可运行该程序，其运行界面如图 1-10 所示。

在 C 语言中，上述程序的执行过程如图 1-11 所示。

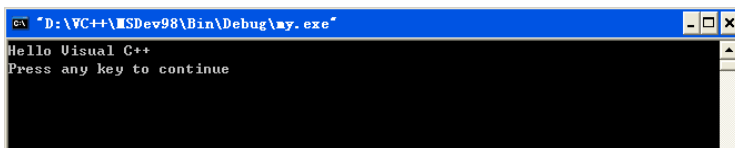


图 1-10 程序运行界面

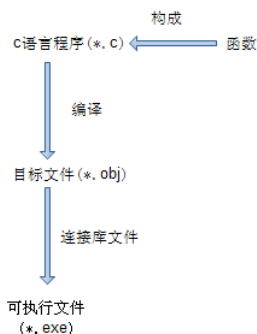


图 1-11 C 语言程序的执行过程



注意：C 语言源文件 (*.c) 经编译只会得到扩展名为*.obj 的目标文件，此过程主要是检查程序是否有错误，若程序有误，则在下方会弹出错误和警告。目标文件必须经过链接生成扩展名*.exe 的文件才能直接在目标机器上运行。

如上述方法创建一个文件，输入下面范例 1.1 的程序，运行观察结果看看。

【范例 1.1】 利用 printf() 函数进行简单的输出。

分析：printf() 函数包含在 stdio.h 头文件中，该函数可以用来实现数据的输出，即将结果输出至屏幕。在使用函数之前，必须将 stdio.h 头文件包含进去。

范例 1.1 代码实现

```

01 #include <stdio.h>          /*包含 stdio.h 头文件*/
02 void main()                /*主函数 main() 入口*/
  
```



```
03 { /*用方括号将下面的代码都包含在 main()函数中*/
04 printf("Hello,My First C Program\n"); /*调用 printf()函数输出一串字符*/
05 } /*右方括号,表示函数体的结束*/
```

【代码分析】本程序是一个简单的 if 应用范例，详细代码分析如下：

- 第 2~5 行是程序要调用的主函数即 main()函数。每个程序都必须要有主函数而且只能有一个主函数。主函数是整个程序的入口，必不可少。
- 第 4 行调用 stdio.h 头文件中的 printf()函数输出“Hello, My First C Program”这句话。

【运行结果】该程序的执行结果如图 1-12 所示。

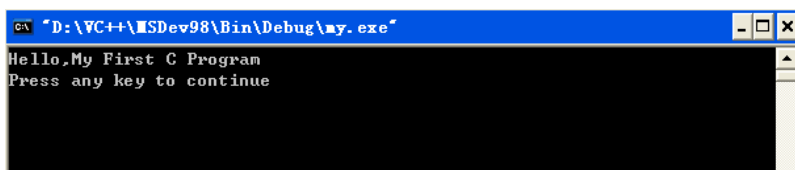


图 1-12 范例 1.1 结果图



注意：在调用 printf()函数之前要把 stdio.h 头文件给包含进去，否则程序会出错，提示找不到 printf()函数。



1.3 良好的代码规范

一个好的程序，代码规范很重要，就像一个人写字的书法一样。若写得很好，别人看起来会感觉很舒服，若很差，别人看懂代码会很费劲，而且管理代码也较头痛。在本节中将讲述如何进行规范代码的书写。

1.3.1 规范命名

在 C 语言中一般被命名的名字都称做标识符。标识符是指一个字符组成的序列，通常包括变量名、常量名、函数名、程序名等。这些名字的命名都必须符合 C 语言的规范，否则程序运行时会出现错误。

命名时必须符合以下 C 语言的规定：

- C 语言中是严格区分大小写的，例如“a”和“A”分别表示两个不同的命名，意义完全不一样。
- C 语言命名需以下画线或字母开头，不能以数字开头，如 0a4、52 都是错误的命名。
- C 语言中命名的名字长度不限，但一般只有前 8 位有效，不同的命名前 8 位一定要不相同。

C 语言中标识符可分为 3 类：

(1) 关键字：指 C 语言中有固定含义的标识符，不能表示其他的含义，包含以下 32 个：

```
auto, extern, register, static, typedef, char, const,
double, enum, float, int, long, short, signed,
```



```
struct, union, unsigned, void, volatile,  
break, case, continue, default, do, else, for, goto,  
if, return, swiTCh, while, sizeof
```



注意：关键字必须是小写的，而且不能拼写错误。

(2) 特定字：指C语言中有特定含义的标识符，不能表示其他的含义，与关键字并无很大区别。

特定字包含：

```
include, define, under, ifdef, ifndef, endif, main
```

(3) 用户标识符：指用户自己定义的一些标识符，如程序中的变量名和函数名等。



注意：用户标识符由下划线、数字、字母等组成，首字符不能为数字，命名的名字不能与关键字和特定字冲突。

1.3.2 美观对称

每个程序有了规范的命名，还必须有规范的排版。这就像一个人写文章一样，要想写好，不仅要有好的书法，还需要有好的排版样式。

在C语言中，代码讲究规范、对称和美观。通常从一个程序中就可以看出一个程序员的编程风格，好的程序员写的代码都很简洁、美观和对称。因此刚开始学习C语言时必须注意养成良好的编程习惯。

建议如下：

(1) 空行。空行虽然不会浪费内存，但浪费纸张。因此需根据实际情况来判断是否需加空行，必要时应加上空行。例如两个函数之间加空行，可以使程序更加清晰。对于前后联系较紧密的语句，则不应加空行。示例如下：

```
void a()  
{  
    ...  
}  
//空行  
void b()  
{  
    ...  
}
```

(2) 一行代码最好只做一件事，不要都挤在一行。例如只定义一个变量、只输出一个语句等。

(3) 在定义变量时就该变量初始化，可以避免变量未初始化引发的问题。

(4) 编译代码时，“{”和“}”要对齐，可使程序简洁。尤其是程序中出现多对“{}”符



号时，对齐的效果非常明显。

(5) 修饰符应紧靠变量，不容易使人产生误解。例如：

```
char* a,b;  
char *a,b; /*不容易误解*/
```

上例中 `char` 和 “*” 一起，很容易让人误解为变量 `a`、`b` 都为 `char` 的指针类型。实际上只有变量 `a` 是 `char` 的指针类型，变量 `b` 是 `char` 类型。

1.3.3 合理注释

在 C 语言代码中添加注释，可用来对程序进行分析说明及提示需注意的问题。注释不会影响程序的执行，在 C 语言中用 “/* */” 来表示。注释虽然有助于代码的理解，但是也不能滥用。

注释建议如下：

(1) 一些简单的语句不用加注释，如下面的代码注释则显的多余。

```
i--; /*i 减 1*/
```

(2) 注释应与源代码相近，不可太远，放在代码的左边、右边、上边都可以。

(3) 注释应适量，不可太多，毕竟注释只起辅助的作用。

(4) 修改代码时，应修改注释，保存代码和注释的一致性。

(5) 注释应尽可能的准确、简洁。

(6) 对于结构化的程序，应在该结构的开头或末尾加注释，便于理解和阅读。

【范例 1.2】用 C 语言打印出简单的 C 字母。

分析：可以先在纸上画出图形，利用 `stdio.h` 头文件中的 `printf()` 函数进行分行输出，最终在屏幕上输出一个字母 C。

范例 1.2 代码实现

```
01  #include <stdio.h>  
02  void main() /*主函数入口，void 表示该函数没有返回值*/  
03  {  
04      printf("****\n"); /*利用 printf() 函数进行相应的输出*/  
05      printf("*\n");  
06      printf("*\n");  
07      printf("****\n");  
08  }
```

【代码分析】本例用 `printf()` 函数进行了简单的输出，详细代码分析如下：

- 第 4~7 行利用 `printf()` 函数进行相应的输出并换行来打印字母 C，其中 `printf()` 函数中的 “\n” 表示换行，输出的内容需要用引号包围起来。

【运行结果】该程序的执行结果如图 1-13 所示。

【范例 1.3】从键盘中输入一个数，并将其输出至屏幕。



分析：利用 `stdio.h` 头文件中的 `scanf()` 函数实现数字的输入，再利用 `printf()` 函数输出该数。

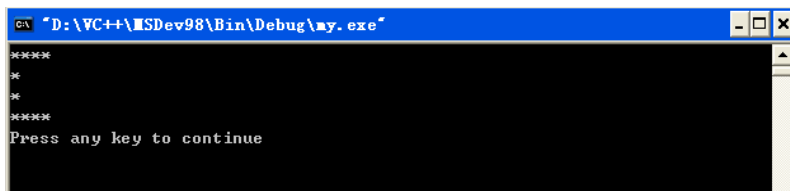


图 1-13 范例 1.2 结果图

范例 1.3 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int a;                      /*定义一个整型变量 a*/
05      printf("input number:");    /*利用 printf() 函数输出提示语句*/
06      scanf("%d",&a);             /*通过 scanf() 函数从键盘获取数据*/
07      printf("The number is:%d\n",a); /*通过 printf() 函数输出变量 a 的值*/
08  }
```

【代码分析】本例利用 `scanf()` 函数和 `printf()` 函数分别进行数据的输入输出，详细代码分析如下：

- 第 3 行定义一个整型变量 `a` 用来保存输入的数字，这种数据类型将会在本书后面的章节讲解到。
- 第 5 行调用 `scanf()` 函数实现数字从键盘输入，将输入的数字保存至相应的变量，即变量 `a` 中。

【运行结果】该程序的执行结果如图 1-14 所示。

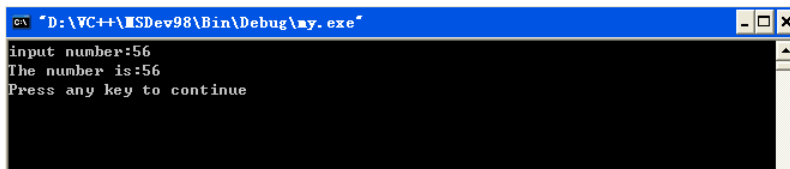


图 1-14 范例 1.3 结果图



注意：`scanf()` 函数中的 “&” 一定要有，否则输入的数字将不能保存至相应的变量中，程序运行结果会出错。



1.4 小结

本章讲解了以下几方面的内容：

- C 语言的一些基本概念以及其开发环境。
- C 语言程序编译运行的基本过程。



由浅入深学 C 语言——基础、进阶与必做 430 题

- 程序的代码规范。
- VC++ 6.0 工具的使用。

这些都是 C 语言的基础，对 C 语言的学习起着很大的作用，读者应认真掌握。



1.5 习题

一、选择题

1. 以下程序正确的是（ ）。

A.

```
#include <stdio.h>
{
    printf("A\n");
}
```

B.

```
void main()
{
    printf("A\n");
}
```

C.

```
#include <stdio.h>
void main()
{
    printf("A\n");
}
```

D.

```
#include <stdio.h>
void mian()
{
    printf("A\n");
}
```

二、填空题

1. 补全下面的程序，让其完整。

```
_____ <stdio.h>
void _____()
{
    int a=1;
    _____("%d\n",a);
}
```

【提示】程序的开头要用#include 命令包含相应的头文件，每一个程序的运行从主函数 main() 入口开始，printf()函数可以用来输出变量的值。

三、编程题

1. 参照例题，编写一个程序输出如下所示的一个倒三角形。

```
*****
*****
*****
*****
*****
*****
***
**
*
```



【提示】利用 `printf()` 函数按照图形进行分行的输出即可。

【核心代码】

```
printf("*****\n");
printf(" *****\n");
printf(" *****\n");
...
```

2. 参照范例 1.3, 从键盘输入一个数, 输出该数的相反数。

【提示】可以在输出该数之前加一个负号, 再输出该数字, 即为该数的相反数。

【核心代码】

```
int x;
scanf("%d",&x);
printf("%d",x);
```

3. 编写一个程序, 实现两个数的相加并将结果输出至屏幕。

【提示】利用 `scanf()` 函数获取从键盘输入的两个数, 输出两个数相加的结果。

【核心代码】

```
int a,b,c;
scanf("%d%d",&a,&b);
c=a+b;
printf("%d",c);
```



注意: `printf()` 函数中的 `c` 可直接用 `a+b` 替换, 这样可省去 `c=a+b` 这一语句, 使程序更简洁。`scanf()` 函数中输入两个数时, 必须用空格或逗号分开, 这样才能区分两个数。

4. 从键盘输入一个数, 判断其是否能被 3 和 5 整除, 若能整除则输出 “yes”, 否则输出 “no”。

【提示】从键盘输入一个数, 可以利用 `scanf()` 函数来实现。然后通过 `if` 语句判断该数能否整除 3 和 5, 若能整除输出 “yes”, 否则输出 “no”。

【核心代码】

```
scanf("%d",&x)
if(x%3==0&&x%5==0)
printf("yes\n");
else
printf("no\n");
```

5. 从键盘输入一个数, 通过 `printf()` 函数输出该数的 2 倍数。

【提示】从键盘输入一个数, 可以通过 `scanf()` 函数将该数保存至一个变量中。然后输出变量的 2 倍值大小。

【核心代码】



```
scanf("%d",&x);  
printf("%d\n",2*x);
```

6. 从键盘输入两个数，求这两个数的乘积并输出至屏幕。

【提示】定义两个变量，保存从键盘输入的数据。另外定义一个变量保存其运算结果，调用 printf() 函数输出结果。

【核心代码】

```
scanf("%d%d",&x,&y);  
z=x*y;  
printf("%d",z);
```

7. 编写一个程序，判断一个数是否为偶数。

【提示】判断一个数是否为偶数，可以通过该数除以 2 的余数来判断。若余数为 0 则为偶数，若余数不为 0 则为奇数。

【核心代码】

```
scanf("%d",&n);  
if(n%2==0)  
printf("该数为偶数\n");  
else  
printf("该数为奇数\n");
```

第 2 章 C 语言基础

要学好一门语言就必须先学好其基础知识，当然 C 语言也不例外。在第 1 章中已经初步讲解了 C 语言的一些概念，本章将会重点讲解程序设计语言的基础知识，以及 C 语言发展历程及其编译和执行过程等方面的内容。

本章主要涉及的知识点有：

- C 语言的发展历程；
- C 语言的特点；
- C 语言程序的编译和执行；
- 流程图；
- 程序语言的种类、特点。



2.1 程序语言基础知识

机器语言是一种低级语言，即计算机所固有的语言，由一串的 0 和 1 的数字来表示，很容易出错。用机器语言写出来的程序很复杂，可读性较差，难以维护和修改。后来人们开始改用助记符来表示操作数和操作码，例如用字母代替 0 和 1 组成的一串数码，汇编语言就这样产生了。汇编语言依旧是一种低级语言，与计算机的指令有着很大的关系，严重限制人们的思想和交流。于是，高级语言应运而生了。高级语言是一种接近人们日常习惯的程序设计语言，由于其易学性、通用性强，便于推广和交流，现被广泛使用。

目前，高级语言有很多种，包括 FORTRAN、BASIC、C、C++ 等 250 多种高级语言。

FORTRAN 是“FORMula TRANslator”（福传）的缩写，也称为“公式翻译器”。它是世界上出现最早的高级程序设计语言，在科学和工程计算等领域起着很大的作用。

C++ 语言是在 C 语言的基础上发展而来的，是一款强大的面向对象程序设计语言，应用于计算机科学的各个领域。它把人们要研究的事物当做一个对象，其中包括状态和操作两个方面。C++ 与 C 相比，主要特点体现在以下两个方面：

- （1）在原有的面向过程机制上，对 C 语言做了很多扩充，例如加入类等功能。
- （2）增加了面向对象的机制。



2.2 C 语言简介

学习一门语言必须要先了解这门语言的概念及其发展历程，本节将会讲解 C 语言的概念及其发展历程。



2.2.1 C 语言发展史

早期的操作系统一般都是用汇编语言编写的，例如 UNIX 系统。由于汇编语言与计算机硬件有关，因此程序的可读性和可移植性都较差，于是人们想用高级语言来编写系统软件。但一般的高级语言不能实现汇编语言的全部功能，例如汇编语言可对计算机内存地址、位等进行操作，一般的高级语言并不能实现。于是人们开始寻找一门语言，要求既有高级语言的特性，又有低级语言（汇编语言）的特性，渐渐的 C 语言产生了。

C 语言既可以用来编写应用软件，也可以用来编写系统软件。

C 语言是以 B 语言为基础发展起来的，它的根源是 ALGOL 60。ALGOL 60 是在 1960 年出现的，但它难以用来编写系统软件，这给人们带来了一定的困扰。1963 年英国剑桥大学在 ALGOL 60 的基础上开发了 CPL（Combined Programming Language）语言，但其规模较大，难以实现。1967 年，剑桥大学对 CPL 语言做了简化，提出了 BCPL（Basic Combined Programming Language）语言。1970 年美国贝尔实验室的 Ken Thompson 对 BCPL 又做了进一步的简化，使其能在 8K 内存中运行，简称为 B 语言。1972 年至 1973 年，D.M.Ritchie 在 B 语言的基础上开发出了 C 语言。C 语言保存了 BCPL 和 B 语言的优点，同时也克服了它们的缺点。后来，C 语言经过多次的改进，越来越完善，功能越来越强大。随着 UNIX 系统的广泛使用，C 语言使用得也越来越广泛。

2.2.2 C 语言特点

C 语言作为一门功能强大的语言，具有很多的特点，其特点如下：

1. 简单、方便灵活

C 语言中包含 9 种控制语句，32 个关键字，书写没有很大的限制，有着很大的自由空间。C 语言既可以作为高级语言来编写程序，也可以对位、字节、地址等计算机基本的工作单位直接进行操作。

2. 运算符种类丰富

C 语言运算符的种类很多，总共包括 34 种运算符。在 C 语言中，括号、类型转换等都会被当做运算符来处理，从而使得 C 语言运算符种类丰富，充分地利用这一特性可以完成其他语言难以实现的运算。

3. 数据结构类型丰富

C 语言中的数据类型包括整型、浮点型、字符型、数组型、指针型、结构体类型和共用体类型 7 种类型，可以用来实现各种复杂的运算，其中指针使得运算的效率更高。

4. 绘图功能强大

C 语言中有专门的库文件，用来实现绘图的功能。它支持多种的显示器和驱动器，也支持多种机型，有很强大的绘图功能。



5. 结构式语言

C语言是一种结构式的语言。它把数据和代码分隔,即程序的各个部分相互独立,除非有必要的交流。这种方式可以使程序层次分明,便于修改、调试和维护。C语言把代码封装在库文件的函数中,直接供用户使用,使程序完全实现结构化。

6. 限制不严格

一般的高级语言对程序的限制都很严格,而C语言中对语法的限制不是很严格,大大方便了用户程序的编写及运行。

7. 生成代码质量高,执行效率高

C语言源文件在编译生成目标文件时,首先会检查是否有错误,若无错误,则会对代码实现优化,才生成目标文件,一般只比汇编语言生成的目标文件代码效率低10%~20%,同时生成的可执行文件执行的效率也很高。

8. 可移植性好

C语言适合于多种操作系统,例如DOS、UNIX和XP等系统,也适应于多种机型,可移植性较好。

C语言也有一些缺点,主要体现在数据的封装性上。它对数据的封装性不强,因此在数据的安全性上有很大的缺陷。

2.2.3 C语言结构

在2.2.2节中讲到了C语言是一门结构式语言,其结构也具有很多特点,如下所示。

- ① 一个C语言源程序可以包含一个或多个源文件。
- ② 一个源文件中可以包含一个或多个函数。
- ③ 一个源程序必须有且只有一个主函数,即main()函数。
- ④ 源文件可以包含预处理命令(#include、#define)。
- ⑤ 每个语句结束时用“;”表示结束,预处理命令和函数的花括号后不用加分号。
- ⑥ 关键字和标识符中间都必须加一个以上的空格用以区分,否则区分不开。



2.3 C程序举例及其构成

本节中举出几个范例,先对C语言形成一个感性的认识。

【范例2.1】在显示器上输出字符串“I am a student”。

分析:要输出语句,可以调用printf()函数进行输出,即将要输出的语言用引号包围在printf()函数中。

范例2.1 代码实现

```
01  #include <stdio.h>           /*包含stdio.h头文件*/
02  void main()                  /*主函数main()*/
```




```
03 {  
04     printf("I am a student\n"); /*通过 printf()函数输出 “I am a student” 语句*/  
05 }
```

【代码分析】本例是 printf()函数的一个简单应用，详细代码分析如下：

- 第 4 行调用 printf()函数输出 “I am a student” 语句，后面的 “\n” 表示换行。

【运行结果】该程序的执行结果如图 2-1 所示。

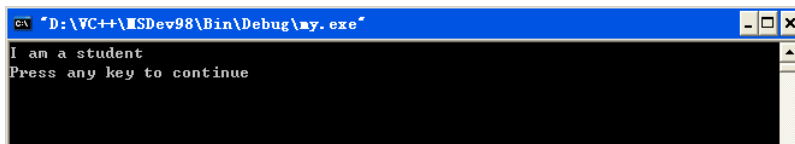


图 2-1 范例 2.1 结果图

【范例 2.2】从键盘输入三个学生的考试成绩，计算其平均分并输出。

分析：定义三个变量保存考试成绩，计算平均分输出即可。

范例 2.2 代码实现

```
01 #include <stdio.h>  
02 void main()  
03 {  
04     float a,b,c,d; /*定义 4 个浮点型变量*/  
05     scanf("%f%f%f",&a,&b,&c); /*通过 scanf()函数实现数据的输入*/  
06     d=(a+b+c)/3; /*计算 3 个变量的平均值并保存至变量 d 中*/  
07     printf("average=%f\n",d); /*调用 printf()函数输出 3 个数的平均值*/  
08 }
```

【代码分析】本例利用 C 语言进行了简单的算数运算，详细代码分析如下：

- 第 4 行定义了 4 个浮点型变量 a、b、c、d，浮点型将会在后面的章节中讲到。
- 第 5 行用 scanf()函数进行数字的输入。
- 第 6 行计算三个数的平均值，并把其保存至变量 d 中。

【运行结果】该程序的运行结果如图 2-2 所示。

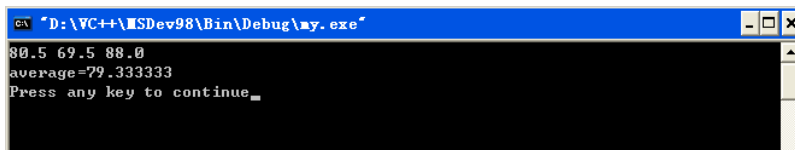


图 2-2 范例 2.2 结果图



注意：scanf()函数中变量参数前必须加上“&”符号，表示将数据写入变量地址中，否则运行结果会出错。

【范例 2.3】输入一个角度 x，计算 x 对应的正切值。

分析：求一个数的正切值，首先须将其转化为对应的角度，然后调用 tan()函数便可求得这



个数的正切值。

范例 2.3 代码实现

```
01  #include <math.h>
02  void main()
03  {
04      float x;                /*定义一个浮点型变量 x*/
05      scanf("%f",&x);          /*调用 scanf() 函数进行数据的输入*/
06      x=(3.14159*x)/180;      /*将数字 x 转换为对应的弧度*/
07      printf("tan(x)=%f\n ",tan(x)); /*调用 tan() 函数计算 tan(x) 的值*/
08  }
```

【代码分析】本例是一个简单的运算范例，详细代码分析如下：

- 第 6 行，把输入的 x 转换为相应的弧度。
- 第 7 行，调用 $\tan()$ 函数计算 $\tan(x)$ 的值，并输出结果。

【运行结果】该程序的运行结果如图 2-3 所示。

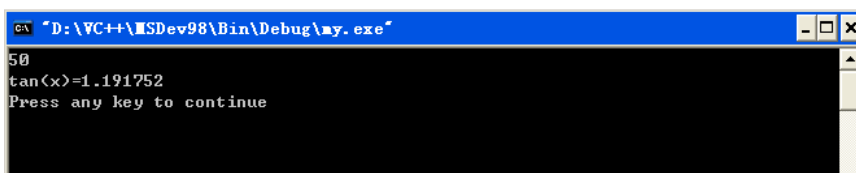


图 2-3 范例 2.3 结果图

【范例 2.4】输入两个数，输出其中的大者。

分析：比较两个数的大小，可以单独写成一个函数，然后在主函数中调用自定义函数，将两个数作为参数比较其大小，最后输出比较结果。

范例 2.4 代码实现

```
01  #include <stdio.h>
02  int max(int a,int b)        /*用户自定义函数 max()*/
03  {
04      int c;                  /*定义一个整型变量 c*/
05      if(a>b) c=a;            /*若变量 a 的值大于 b，将变量 a 赋值给变量 c*/
06      else c=b;               /*若变量 a 的值小于或等于 b，将变量 b 的值赋给变量 c*/
07      return(c);              /*函数调用结束时返回变量 c 的值*/
08  }
09  void main()
10  {
11      int x,y,z;              /*定义 3 个变量 x、y 和 z*/
12      scanf("%d%d",&x,&y);     /*调用 scanf() 函数实现数据的输入*/
13      z=max(x,y);             /*调用 max() 函数比较 x 和 y 值的大小，并将大者赋给变量 z*/
14      printf("max=%d\n",z);
15  }
```

【代码分析】本示例为如何对两个数进行比较，详细代码分析如下：



由浅入深学 C 语言——基础、进阶与必做 430 题

- 第 2~8 行为用户自定义函数 `max()` 函数，用来比较两个数的大小。
- 第 9~15 行，在主函数中调用自己编写的函数进行两个数大小的比较，再输出其中的大者。

【运行结果】该程序的运行结果如图 2-4 所示。

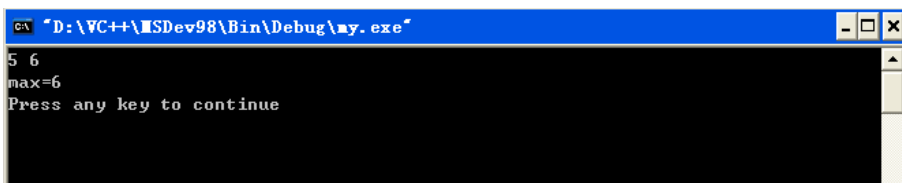


图 2-4 范例 2.4 结果图



注意：本程序中有两个函数，它的执行顺序不是从上至下的，是从 `main()` 函数开始执行的，并在 `main()` 函数中调用自定义的函数。

从上面例题中，可以看出 C 语言程序结构的基本形式如下：

```
#include <文件>          /*可包含多个文件*/
类型 main()
{
    操作
}
/*自定义函数*/
函数类型 函数名()
{
    函数体
}
```

`#include` 是一种预处理命令，可以把一些文件包含至程序中。例如 `#include <stdio.h>`，是把 `stdio.h` 头文件包含进程序中。

C 语言程序由主函数和其他函数组成，其中函数是 C 语言的基本单元。

主函数是整个程序的入口，即每个程序运行都是从主函数开始执行的。主函数用 `main()` 表示，可以出现在程序的任何位置。

其他函数包括库函数和用户自定义函数。库函数指系统已经定义好的并保存在相应文件中的函数，例如上面例子中用的 `scanf()` 函数、`printf()` 函数都是库函数，保存在 `stdio.h` 头文件中，在调用之前必须用 `#include` 语句将相应的头文件包含进来。用户自定义函数是指用户自己编写的函数，如范例 2.4 的 `max()` 函数。

C 语言中用 `“/* */”` 表示注释，程序编译运行时，会忽略掉注释。注释可使程序的可读性大大提高，便于修改和维护。

同时在 C 语言程序书写时应注意以下三个方面：

- (1) 尽量一行一个语句，使程序简洁。
- (2) C 语言中一般使用小写字母，严格区分大小写。
- (3) 注意代码的缩进，使程序对齐美观。

2.4 C程序的编译和执行

一个源文件运行得出结果，都必须经过编译和运行的过程。C语言程序的执行过程一般如图2-5所示。

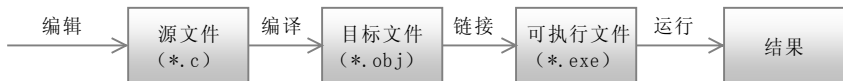


图 2-5 C语言执行过程

本节将会讲述C语言编译和执行的整个过程。

2.4.1 编译程序

编译的功能：读取源文件，对其进行词法和语法分析，把高级语言转换为与之功能相同的汇编代码，接着由汇编代码转换成机器代码，并根据机器的要求生成相应的可执行程序。其过程很复杂，可分为以下6个阶段：

- (1) 词法分析阶段。词法分析根据语言的词法（单词结构）规则来分析，词法的规则可用正规文法或正规式来表示，是指有限自动机能识别正规文法的语言和正规式组成的集合。
- (2) 语法分析阶段。语法分析在词法分析的基础上将单词分别分解成各类语法单位，它依据语言的语法规则（程序结构规则）对源程序结构进行分析。
- (3) 语义分析阶段。这一阶段审查代码是否有语义错误，为代码生成阶段做准备。
- (4) 中间代码生成阶段。经过上述阶段后，会生成一个中间代码，即把源程序转化为一种内部的记号语言。这一阶段依据的是中间代码生成规则。
- (5) 代码优化阶段。代码优化阶段是指对中间代码进行改造和优化，使得代码节省空间和时间，提高执行效率。
- (6) 目标代码生成阶段。目标代码生成阶段把中间代码转化为机器上的指令代码，这一过程生成的目标代码与机器硬件和系统有关。

一个源文件经过以上的6个过程即可转化为目标文件代码，即把高级语言转化为机器语言。

C语言编译的完整过程如图2-6所示。

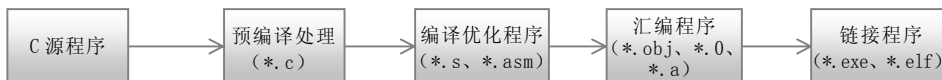


图 2-6 C语言编译全过程

1. 编译预处理

读取C源文件，处理伪指令和特殊符号。伪指令一般指以“#”为开头的指令，包含以下4个方面：

(1) 宏定义指令

宏定义又称为宏替换，指用一个表达式替换另一个式子，简称为宏。例如#define name Tom,



预编译处理该指令即把 `name` 用 `Tom` 来替换。当程序中不需要替换时，可以 `#undef` 取消宏的定义，宏定义取消后 `name` 不会被 `Tom` 替换了。

(2) 条件指令

条件指令可使程序员通过定义不同的宏自由控制程序代码的执行，执行哪些代码，跳过哪些代码。预编译处理会将那些多余的代码去掉。

(3) 头文件指令

头文件指令，如 `#include<文件>`，是指把相应的头文件包含至程序中。包含头文件，是为了包含头文件中系统定义好的函数，以便供程序直接调用。例如 `#include <stdio.h>` 是指把 `stdio.h` 包含进来，该文件中包括 `printf()` 和 `scanf()` 等函数。

包含的 C 语言中的头文件既可以是系统提供的，也可以是用户自定义的文件。若是系统中的头文件，在用 `include` 引入时用 “`<>`” 括起来，若是用户自定义的头文件，用 “`”` 包围，且要把相应的头文件放到当前的工作目录中。

(4) 特殊符号

预编译处理时会检测有无特殊符号，例如 `LINE`、`FILE` 等都会被当做特殊符号处理，用适当的值来替换。

程序源文件经过预编译处理后，会变成一个没有宏定义、条件指令、头文件指令和特殊符号的文件。这个文件功能与源文件是一样的，但内容有所改变。

2. 编译和优化

经过预编译得到的文件，只包含数字、字符串、变量和 C 语言的关键字，不再包含其他内容。

- 编译：把预编译得到的文件进行词法、词义、语法等分析，将其转化为与之功能等价的中间代码或汇编代码。
- 优化：对中间的代码进行优化，使其时间复杂度和空间复杂度变小，即执行效率变高。

3. 汇编程序

一个程序要被执行，最终都要被转化为机器语言。汇编程序是把汇编代码转化为机器代码，即机器所能识别的语言。

4. 链接程序

经过汇编程序生成的目标文件可能还存在一些问题，例如一个文件要调用另一个文件中的函数，或调用库文件中的函数。要解决这些问题，必须要经过链接的过程。

链接程序把彼此有关的目标文件都联系起来，即把这些文件都联系成一个统一的整体装入系统的内存中，使目标文件能够相互调用。

C 语言的编译全过程很复杂，其中的具体过程可能还与计算机的硬件配置有关，读者只要知道其分为编译和链接两个过程就可以了。其中编译时产生后缀名为 `(.obj)` 的文件，链接则产生后缀名为 `(.exe)` 的文件。

2.4.2 解释程序

C 语言是通过编译程序实现运行的，但有些语言则通过解释程序来实现，例如 BASIC 语言就是通过解释程序来实现运行的。



编译程序与解释程序有很大的区别。编译程序读取整个文件并把其转化为目标文件从而实现运行，解释程序不会把程序转化为目标代码，它是通过一行一行地读取程序来实现相应操作的。

一般来说，编译产生的代码运行时间会比解释程序短。编译程序较复杂，维护和开发都较困难。相比较而言，解释程序较简单，但其运行时间较长。

2.4.3 分块编译

一般编写很短的C语言程序时，一个文件就足够了。当编写的代码很长时，就需要把它们分成多个块，分别在不同的文件中进行编写，对每个文件分别进行编译然后连接起来，最终形成一个目标文件运行得出结果。分块编译可以单独修改一个文件的代码，而不会影响其他文件的代码。

【范例 2.5】编写一个程序，实现分块进行编译。

分析：当程序很复杂时，可以将代码分别放在不同的文件中进行编写。这样可使程序简洁，同时使得程序的维护和修改更简单。

范例 2.5 代码实现

```
01  #include <stdio.h>
02  #include <cfile.h>          /*包含用户自定义 cfile.h 头文件*/
03  void main()
04  {
05      int a,b,c;              /*定义 3 个整型变量 a,b,c*/
06      a=5;                    /*将 5 赋值给变量 a*/
07      b=9;                    /*将 9 赋值给变量 b*/
08      c=fun(a,b);             /*调用 fun() 函数并将其返回值赋给变量 c*/
09      printf("%d\n",c);
10  }
```

cfile.h 代码如下：

```
01  int fun(int x,int y)        /*自定义函数 fun(), 其中 x 和 y 作为形参*/
02  {
03      return y-x;              /*返回变量 y 减去 x 的值*/
04  }
```

【代码分析】本示例为如何对程序进行分块编译，详细代码分析如下：

- 第 2 行，包含了用户自定义文件，即 cfile.h 头文件。
- 第 5 行，定义了 3 个整型变量 a、b、c。
- 第 6~7 行，将值 5 和 9 分别赋值给变量 a 和 b。
- 第 8 行，调用 fun() 函数，其中 a 和 b 作为参数传递给函数，变量 c 用于保存从函数返回的计算结果。

【运行结果】该程序的运行结果如图 2-7 所示。

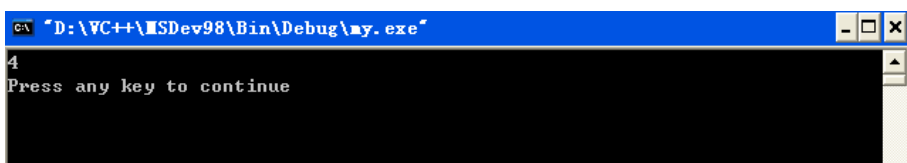


图 2-7 范例 2.5 结果图

2.4.4 函数和连接

C 语言程序本质上就是一个函数，即主函数，在主函数中调用其他函数来实现相应的功能。

C 语言为用户提供了一个标准库。当 C 语言中出现未定义的函数时，程序就会在相应的库文件中寻找与之对应的函数并调用，但调用函数之前，必须要先用“`#include`”语句把对应的库文件包含至程序中。

【范例 2.6】编写一个程序，调用自定义函数实现两个数的相加。

分析：编写一个函数实现两个数的相加，则需将两个数作为参数传递给函数，在函数中进行相加，再将结果返回至主函数中，最后输出即可。

范例 2.6 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  int fun(int x,int y)                                /*自定义函数 fun()*/
03  {
04      return(x+y);
05  }
06  void main()
07  {
08      int a=4,b=5,c;                                  /*定义整型变量*/
09      c=fun(a,b);                                     /*调用 fun() 函数*/
10      printf("%d\n",c);
11  }
```

【代码分析】本示例为如何调用自定义函数，详细代码分析如下：

- 第 2~5 行，为自定义函数 `fun()`。该函数计算两个数的累加和，并将其值返回。
- 第 9 行，调用 `fun()` 函数计算两个数的累加和，其中 `a` 和 `b` 作为参数传递给函数。函数使用方法及其参数将会在后面的章节中讲解到。

【运行结果】该程序的运行结果如图 2-8 所示。

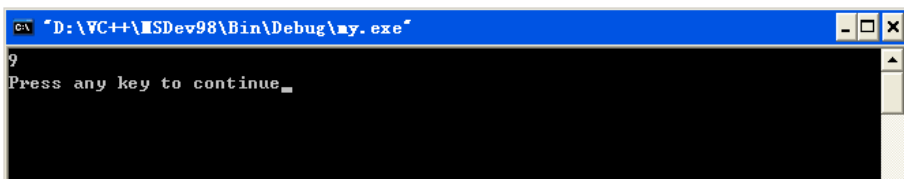


图 2-8 范例 2.6 结果图



2.4.5 运行程序

运行是指将最后生成的可执行文件运行，得出程序的结果。若程序运行的结果不正确，可以对源文件重新进行编辑，对其编译链接再运行观察结果，直至得出正确结果。运行程序与编译和链接不同，可以在编译环境中运行，也可以脱离环境直接在机器上运行。



2.5 算法设计与分析

算法是指人们解决实际问题时采用的方案，其在解决问题的过程中起着至关重要的作用。在本节中将讲解算法的概念、特性及其应用。

2.5.1 算法简介

算法是程序的灵魂，一个好的程序必然有着一个好的算法，可见算法对程序的设计是很重要的。算法是指求解问题的步骤，包含有限个字符和操作，数学公式、伪代码等都可以看做算法。

算法有如下特性：

- ① 算法有 0 个或多个外部量输入。
 - ② 算法至少输出一个量。
 - ③ 算法的每条指令都需清晰、无歧义
 - ④ 算法的执行时间必须有限，即指令执行次数有限，执行每条指令的时间也必须有限。
- 程序是用一种语言及算法实现的。它可以不满足上述算法特性的第 4 条，即时间的有限性。

【范例 2.7】 已知 $s=1+2+3+\cdots+50$ ，求 s ，设计相应的算法。

分析如下。

算法一：

- (1) 定义一个变量 $s=1$;
- (2) 令 $s=s+2$;
- (3) 令 $s=s+3$;

.....

- (50) 令 $s=s+50$; 最后 s 即为正确的结果，输出即可。

算法二：增加一个变量 i ，用循环方法计算其和。

- (1) 定义变量 s ，令其为 0。
- (2) $s=s+i$; $i=i+1$;
- (3) $i \leq 50$ 就一直执行第二步，若 $i > 50$ 则输出结果即可。

算法三：利用数学公式 $s = (1+50) / 2 \times 50$ 直接计算。

显然在上面三种算法中若采用第三种算法，则很方便就算出结果了。因此一个好的算法对程序至关重要。



一个好的算法不仅要包含算法的特点，还应该具有以下特点：

(1) 正确性。一个算法必须要首先得保证其能正确计算出结果，再来判断算法的好坏。

(2) 可行性。算法要求每一步都能够执行，符合实际的理论，并能得出正确的结果。

(3) 高效性。每一个问题都有多种不同的算法可以实现，这些算法中执行时间最短、最节省空间的就称其为最好的算法。

2.5.2 算法复杂性

算法复杂度分为时间复杂度和空间复杂度，人们经常通过时间复杂度和空间复杂度的高低来判断一个算法的好坏。

1. 时间复杂度

一个程序在机器上运行所花费的时间与很多方面有关系，例如算法的规模、算法用的语言等。如果不管这些因素的影响，光从代码考虑，人们通常用时间复杂度来度量算法运行时间的长短，如时间复杂度在 n 级，记作 $O(n)$ 。

2. 空间复杂度

空间复杂度用来度量算法占用的空间的大小，一般来说占用空间越小，算法就越好。空间复杂度一般用 $o()$ 来表示。例如一个算法的空间复杂度为 n ，则表示为 $o(n)$ 。



2.6 小结

在本章重点讲述了 C 语言的基础知识，其中包含 C 语言发展历程、C 语言编译及链接过程、算法等。对于这些概念，读者只需了解即可，不用做深入的研究。



2.7 习题

一、填空题

1. 以下程序的运行结果为_____。

```
#include <stdio.h>
void main()
{
    int a=5,b=6;
    printf("%d,%d\n",a,b);
}
```

【提示】上述程序中定义了两个变量 a 和 b ，对两个变量进行了初始化，调用 `printf()` 函数输出其结果至屏幕。

2. 以下程序运行结果为_____。



```
#include <stdio.h>
void main()
{
    int x=3,y=4;
    x=y;
    printf("%d,%d\n",x,y);
}
```

【提示】上述程序定义了两个变量 x 和 y ，初始化 x 和 y 的值为 3 和 4，再将 y 的值赋给变量 x ，最终 x 和 y 的值都为 4。

3. 以下程序运行结果为_____。

```
#include <stdio.h>
void main()
{
    int x=3,y=4;
    int t;
    t=x;
    x=y;
    y=t;
    printf("%d,%d",x,y);
}
```

【提示】上述程序中定义两个变量 x 和 y ，初始化其值为 3 和 4。然后通过变量 t 交换 x 和 y 的值。即先用变量 t 保存变量 x 原来的值，将变量 y 的值赋给变量 x ，再将变量 t 的值赋给变量 y ，最后输出两个数交换后的结果。

4. 以下程序运行结果为_____。

```
#include <stdio.h>
void main()
{
    int a=10;
    printf("%d",a-1);
}
```

【提示】上述程序定义一个变量 a ，初始化其值为 10，调用 `printf()` 函数输出变量 a 减 1 的值的大小。

二、编程题

1. 参照范例 2.5，求 1~100 的和。

【提示】本例有多种方法可以实现，其中直接套用数学公式 $s = (1+100)/2 \times 100$ 最方便。

【核心代码】

```
s=(1+100)/2*100;
printf("%d",s);
```

2. 参照范例 2.4，从键盘输入两个数，求出这两个数中的小者并将结果输出，要求比较操



作单独编写一个函数。

【提示】范例 2.4 中输出了两个数中的大者，本题要求输出小者，只需改变一些符号即可。

【核心代码】

```
int min(int a,int b)
{
    if(a>b) c=a;
    else c=b;
    return(c);
}
void main()
{
    scanf("%d%d",&x,&y);
    z=min(x,y);
}
```

3. 从键盘输入两个数，将这两个数交换后输出。

【提示】从键盘输入两个数，可以通过 scanf()函数实现。交换两个数可以编写为另外一个函数，然后在主函数中调用实现交换功能。

【核心代码】

```
void main()
{
    scanf("%d%d",&x,&y);
    f(x,y);
}
f(int x,int y)
{
    t=x;
    x=y;
    y=t;
    printf("%d %d",x,y);
}
```

4. 从键盘输入一个数，编写程序输出该数的绝对值至屏幕。

【提示】调用 math.h 头文件中的 fabs()函数可以求得数的绝对值，最后调用 printf()函数输出结果至屏幕。

【核心代码】

```
scanf("%d",&x);
y=fabs(x);
printf("%d",y);
```

5. 编写程序，从键盘输入两个数，求出其中的大者。

【提示】从键盘输入两个数，然后对这两个数进行比较，输出其中的大者至屏幕。

【核心代码】



```
scanf("%d%d",&x,&y);
if(x>=y)
    z=x;
else
    z=y;
printf("两者中的大者为%d\n",z);
```

6. 编写程序，求 1~100 中的素数。

【提示】素数是指除了 1 和其本身之外不能被其他数整除的数。可以穷举 1~100 之间的数并进行判断，若该数为素数则输出至屏幕。

【核心代码】

```
for(i=1;i<=100;i++)
{
    flag=1;
    for(j=0;j<=i/2;j++)
        if(i%j==0)
            flag=0;
    if(flag==1)
        printf("%d ",i);
}
```

7. 编写程序，从键盘输入一个数并求其阶乘值。

【提示】求一个数的阶乘，可以定义一个变量 i 从该数一直减 1 直到 1 为止，计算其累乘结果输出至屏幕。

【核心代码】

```
scanf("%d",&n);
for(i=n;i>0;i--)
    s=s*i;
printf("%d 的阶乘为%d\n",n,s);
```

8. 编写程序，求 1~100 中所有奇数之和。

【提示】求 1~100 之间的所有奇数之和，可以设置一个变量从 1 变化至 99，同时设置一个变量保存其中奇数的累加和再输出至屏幕，其初始值应为 0。

【核心代码】

```
for(i=1;i<100;i=i+2)
    s=s+i;
printf("%d",s);
```

9. 编写程序，要求从键盘键入 n ，求 $1/3+2/4+3/5+\cdots+n-2/n$ 的值。

【提示】因为上述式子的值为浮点型，因此应将保存结果的变量定义为浮点型，然后控制数从 $1/3$ 变化至 $n-2/n$ 计算其累加和。

【核心代码】

```
scanf("%d",&n);
```



```
i=1;
j=3;
while(j<=n)
{
    s=s+i/j;
    i++;
    j++;
}
```

第3章 变量和数据类型

变量是C语言中的重点，在程序中经常使用，数据的输入、输出一般都是用变量实现的。在现实生活中，人们处理的每个数据都会不同，例如有些数据为整数，而有些则为小数。因此在C语言中，每个变量可以通过定义为不同的数据类型，对不同数据进行操作。

本章主要涉及的知识点有：

- 数据类型；
- 变量；
- 常量；
- 运算符及其表达式；
- 强制类型转换。



3.1 常量及符号常量

在C语言中，常量常用于变量的初始化，其值不能改变。在本节中，将讲解常量及符号常量的概念和应用。

3.1.1 常量

在程序中，有常量和变量之分。

- 常量：程序运行过程中，值不能被改变的量。
- 变量：程序运行过程中，值可以变化的量。

常量区分其类型。例如1、3、50等为整型常量，1.25、-8.32等为浮点型常量（浮点型又可分为单精度和双精度浮点型，将会在后面的内容中讲到），'a'、'z'等为字符型常量。

常量常常被用在初始化变量或其他语句中，示例如下：

```
int a=1;           /*把整型常量赋给a*/  
float b=1.0;       /*把浮点型常量赋给b*/  
char c='f';        /*把字符型常量赋给c*/
```



注意：在把字符型常量赋值给字符型变量时，要用单引号包围起来，如'a'。

【范例 3.1】编写一个程序，定义一个字符变量并进行初始化，将字符变量的值输出至屏幕。

分析：定义一个字符变量时，将字符型常量赋值给字符变量便可实现初始化，再通过 printf() 函数输出字符变量的值至屏幕。



范例 3.1 代码实现

```
01  #include <stdio.h>                /*包含 stdio.h 头文件*/
02  void main()                       /*主函数 main()*/
03  {
04      char c='a';                   /*定义一个字符变量 c 并初始化其值为字符 a*/
05      printf("%c\n",c);             /*调用 printf() 函数输出字符变量 c 的值*/
06  }
```

【代码分析】本例为字符常量的简单范例，详细代码分析如下：

- 第 4 行，定义一个字符型变量 `c`，并初始化其值为字符 `'a'`。

【运行结果】该程序的执行结果如图 3-1 所示。

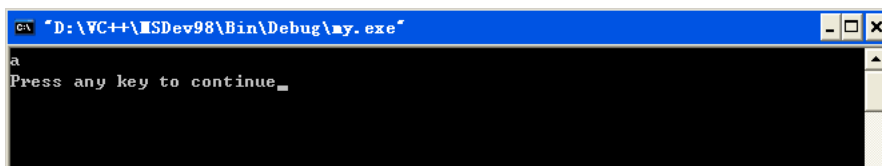


图 3-1 范例 3.1 结果图

3.1.2 符号常量

符号常量是指用一个标识符来代表一个常量。如：

```
#define x 5
#include <math.h>
void main()
{
    float y;
    y=sqrt(x);
    printf("%f",y);
}
```

上述程序中 `x` 代表 5，程序运行过程中凡是 `x` 出现的地方都会被 5 替换。

【范例 3.2】编写一个程序，实现符号常量的简单应用。

分析：符号常量相当于宏定义，在程序运行的过程中标识符都会被相应的常量所替换。

范例 3.2 代码实现

```
01  #include <stdio.h>
02  #define s 8                        /*定义一个符号常量 s*/
03  void main()
04  {
05      int x=5;                       /*定义一个整型变量 x，初始化其值为 5*/
06      printf("%d,%d\n",x,s);        /*调用 printf() 函数输出变量 x 和 s 的值*/
07  }
```

【代码分析】本例为符号常量的简单范例，详细代码分析如下：



- 第2行，定义一个符号常量 `s`，其表示的值为 8。
- 第5行，定义一个整型变量 `x`，初始化其值为 5。

【运行结果】该程序的执行结果如图 3-2 所示。

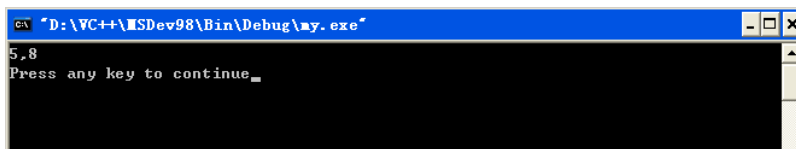


图 3-2 范例 3.2 结果图



注意：符号常量不能被赋值，若赋值则会出错。上述程序中如出现 `s=10;` 则会报错。



3.2 变量

3.2.1 变量的概念及定义

C 语言程序经常使用变量，变量的值在程序运行的过程中可以改变，一般在内存中有一块连续的存储空间。变量是用来保存常量的，就像一个不同大小的仓库，用来存放不同的物品，这些物品则可以看做是常量。

变量的定义主要包含两方面：一个是变量的数据类型，另一个是变量的名称。不同的数据类型变量可用来存放不同类型的数据。变量的名称用来标识不同的变量，可直接用名字引用一个变量，但名字的命名应符合 C 语言的命名标准。

C 语言程序变量定义的形式如下：

```
类型 变量名;
```

一个类型可以定义一个变量，也可以定义多个变量。当定义多个变量时，每个变量之间要用逗号隔开，最后一个变量用分号表示定义语句的结束。例如：

```
int a;  
float b,c,d;
```

同一类型的多个变量定义时，也可分多行定义。如：

```
float b;  
float c;  
float d;
```

很明显，这样没有在一行中定义简洁。



说明：不同的数据类型用不同的关键字表示，例如整型用 `int` 表示，浮点型用 `float` 表示。这一概念将会在后面的内容中讲到。



3.2.2 变量地址

每一个变量定义时，系统都会为其分配一个相应的存储空间，并且会记录下其位置，这一位置称为变量的地址。当程序把一个常数放入变量中，本质上是把常量存储至与变量对应的地址中。当要对这一变量进行修改时，编译系统会找到变量的地址，并改变该地址中的内容。

例如 `s=50`，假设 `s` 的内存单元编码为 3004，则其存储状态如表 3-1 所示。

表 3-1 s 存储状态

3001	3002	3003	3004	3005	3006	3007	3008
			50				

用户在调用变量时，不可能知道其确切的地址。因此 C 语言提供了一个取地址符“&”，可直接对变量的地址空间进行操作，简便了程序的编写。例如：

```
scanf("%d",&s);
```

该语句表示把从键盘输入的数字存储至变量 `s` 中，其中的“&`s`”表示变量 `s` 的地址。

【范例 3.3】 编写一个程序，将变量的值及地址输出至屏幕。

范例 3.3 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x;                /*定义一个整型变量 x*/
05      x=5;                  /*将 5 赋值给变量 x*/
06      printf("%d\n",x);     /*调用 printf()函数输出变量 x 的值*/
07      printf("%d\n",&x);    /*调用 printf()函数输出变量 x 的地址值*/
08  }
```

【代码分析】 本例为变量定义范例，详细代码分析如下：

- 第 4 行，定义了一个整型变量 `x`。
- 第 7 行，通过 `printf()` 函数和 `&` 运算符输出变量 `x` 的存储地址。

【运行结果】 该程序的执行结果如图 3-3 所示。

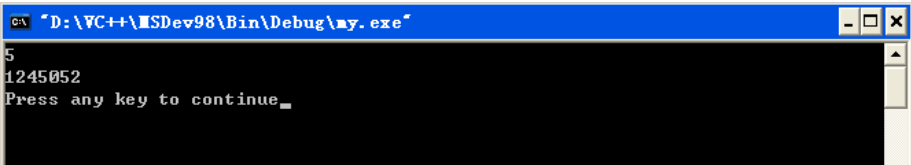


图 3-3 范例 3.3 结果图



注意： & 符号表示取变量的地址，如 `&x` 表示变量 `x` 的地址，而不是变量的值。



3.2.3 变量初始化

变量赋值包括两种方式，一种是先定义变量再赋值给变量，另一种是在变量定义的同时就给变量赋值。后者即称为变量的初始化，C语言中各种类型的变量都可以初始化。例如：

```
int a=1;
float b=1.0;
double c=1.0;
char d='c';
```



注意：变量的初始化与变量的赋值语句是不同的。例如：

```
int a;a=1;
int a=1;
```

虽然上面两种运行的结果是一样的，但其赋值方式是不同的。

【范例 3.4】 编程一个程序，定义一个变量初始化后输出结果至屏幕。

范例 3.4 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int a=10;
05      printf("%d\n",a);          /*输出变量 a 的值*/
06  }
```

【代码分析】 本例为变量初始化范例，详细代码分析如下：

- 第4行，定义一个整型变量a，初始化其值为10。

【运行结果】 该程序的执行结果如图3-4所示。

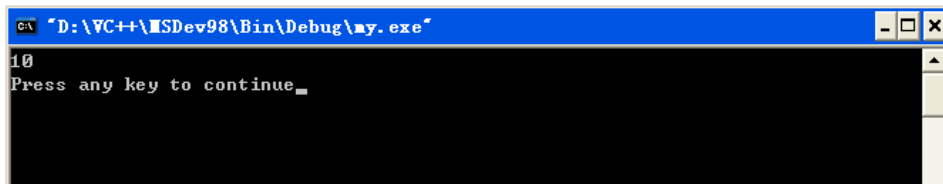


图 3-4 范例 3.4 结果图



3.3 C语言的基本数据类型

一个变量在定义时必须指出其数据类型，C语言提供了丰富的数据类型，可以用来处理多种不同的数据。C语言中的数据类型如表3-2所示。



表 3-2 C 语言数据类型及分类

数据类型	基本类型	整型	短整型（short）	
			整型（int）	
			长整型（long）	
		浮点型	单精度浮点（float）	
			双精度浮点（double）	
	字符型（char）			
	构造类型	数组		
		结构体（struct）		
		共用体（union）		
		枚举类型（enum）		
	指针类型			
	空类型（void）			

不同的数据类型在机器中占据的位数不同，因此存储数据的范围也不同，具体如表 3-3 所示。

表 3-3 数据类型位数及范围

关键字	机器中所占位数	取值范围
char	8	-127~127
signed char	8	-127~127
unsigned char	8	0~255
int	16	-32768~32767
signed int	16	-32768~32767
unsigned int	16	0~65535
signed short int	16	-32768~32767
unsigned short int	16	0~65535
long int	32	-2147483648~2147483647
float	32	$-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
double	64	$-1.7 \times 10^{-308} \sim 3.4 \times 10^{308}$
long double	80	$-3.4 \times 10^{-4392} \sim 3.4 \times 10^{4392}$

3.3.1 整型常量

整型常量就是整数型的常数。它包含三种不同的形式：

(1) 八进制整数：是指以 0 开头由 0~7 组成的数字，如 0234 表示八进制数 234，012 表示八进制数 12，相当于十进制数 10。

(2) 十进制整数：是以非 0 开头由数字 0~9 组成的数字。如 11、782 等。

(3) 十六进制整数：以 0x 开头，并由 0~9 和 A~F 组成的数字。十六进制中的 A~F 表示数字 10~15。例如 0x345，表示十六进制数 345，相当于十进制数 837，其转化过程为 $(345)_{16} = 3 \times 16^2 + 4 \times 16^1 + 5 \times 16^0$ 。



【范例 3.5】 已知两个数分别为 0246 和 0x246，将其转化为十进制数并输出至屏幕。

分析：0246 为八进制数，0x246 为十六进制数。C 语言中提供了数据进制的自动转化，可以定义两个变量保存这两个数，再用十进制输出这两个数。

范例 3.5 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=0246;                /*定义一个整型变量 x，并初始化其值为 0246*/
05      int y=0x246;              /*定义一个整型变量 y，并初始化其值为 0x246*/
06      printf("x=%d,y=%d\n",x,y); /*输出变量 x 和 y 对应的十进制数*/
07  }
```

【代码分析】 本例是整型常量相互转化的简单范例，详细代码分析如下：

- 第 4 行定义了一个变量 x，保存八进制数 0246。
- 第 5 行定义了一个变量 y，保存十六进制数 0x246。

【运行结果】 该程序的执行结果如图 3-5 所示。

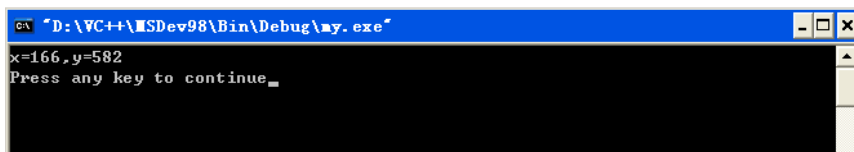


图 3-5 范例 3.5 结果图



注意：

- (1) 在整数的后面可以加“L”，表示该整数为长整型。例如 30L、079L 等。
- (2) 在整数的后面加上“u”或“U”可以表示该数为无符号整数，如 38u、0x45u 等。
- (3) 若数字超过整型的取值范围，则会溢出，此时应选用长整型。

3.3.2 整型变量

整型变量的基本类型为 int 类型，从取值范围和有符号可分为以下 6 类。

(1) int: 有符号整型。字长为 2 字节，占位 16 位（1 个字节占 8 位），取值范围为-32768~32767。在有些机器中，其字节长为 4 字节，占位 32 位。定义语句如下：

```
int a;
```

(2) signed short int: 有符号短整型。字长为 2 字节，占位 16 位，取值范围为-32768~32767。定义语句如下：

```
signed short int a;
```

(3) signed long int: 有符号长整型。字长为 4 字节，占位 32 位，取值范围为-2147483648~2147483647。定义语句如下：



```
signed long int a;
```

(4) **unsigned int**: 无符号整型。字长为 2 字节, 占位 16 位, 取值范围为 0~65535。在有些机器中, 其字节长为 4 字节, 占位 32 位。定义语句如下:

```
unsigned int a;
```

(5) **unsigned short int**: 无符号短整型。字长为 2 字节, 占位 16 位, 取值范围为 0~65535。定义语句如下:

```
unsigned short int a;
```

(6) **unsigned long int**: 无符号长整型。字长为 4 字节, 占位 32 位, 取值范围为 0~4294967295。定义语句如下:

```
unsigned long int a;
```

【范例 3.6】 将不同数据类型的数据转化为十进制数并输出结果。

分析: 定义变量用来保存不同类型的数据, 再利用 C 语言提供的数据进制自动转化机制, 将其用十进制表示并输出即可。

范例 3.6 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=072;                /*定义一个整型变量 x, 初始化其值为 072*/
05      unsigned int y=0x30;      /*定义一个无符号整型变量 y, 初始化其值为 0x30*/
06      long int z=55;            /*定义一个长整型变量 z, 初始化其值 55*/
07      short int u=8;            /*定义一个短整型变量 u, 初始化其值为 8*/
08      printf("%d,%d,%d,%d\n",x,y,z,u); /*调用 printf() 函数输出变量 x、y、z 和 u 对应的
                                     十进制数*/
09  }
```

【代码分析】 本例是一个简单的范例, 详细代码分析如下:

- 第 4~7 行, 分别定义了不同类型的变量来存储不同进制的数字。

【运行结果】 该程序的执行结果如图 3-6 所示。

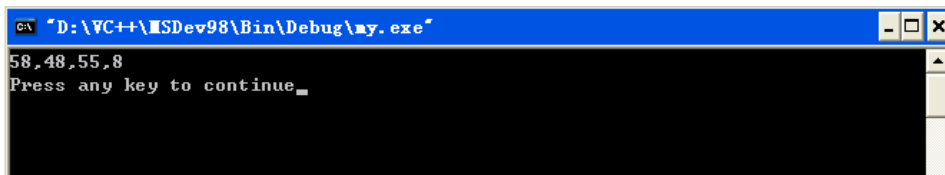


图 3-6 范例 3.6 结果图



注意: 不同的数据类型有不同的数值表示范围, 根据实际情况需定义为不同的类型。



【范例 3.7】根据给出的不同类型的数据，进行相应的+、-、×、/运算，并将其结果输出。

分析：定义相应的数据类型保存不同的数据，对其进行加、减、乘、除操作，最后利用 printf() 函数输出计算结果。

范例 3.7 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x;                /*定义一个整型变量 x*/
05      unsigned int y;       /*定义一个无符号整型变量 y*/
06      int a,b,c,d;         /*定义 4 个整型变量 a,b,c,d*/
07      x=30;
08      y=20;
09      a=x+y;               /*将变量 x、y 累加和赋值给变量 a*/
10      b=x-y;               /*将变量 x 减去 y 的值赋值给变量 b*/
11      c=x*y;               /*将变量 x 和 y 的乘积赋值给变量 c*/
12      d=x/y;               /*将变量 x 除以 y 的值赋值给变量 d*/
13      printf("%d,%d,%d,%d\n",a,b,c,d); /*调用 printf() 函数输出变量 a、b、c 和 d 的值*/
14  }
```

【代码分析】本例为不同类型数据之间运算的简单范例，详细代码分析如下：

- 第 4 行，定义了一个整型变量 x。
- 第 5 行，定义了一个无符号整型变量 y。
- 第 6 行，定义了 4 个整型变量，用于保存加减乘除的结果。
- 第 7~8 行，分别给变量 x 和 y 赋初始值。
- 第 9~12 行，对变量 x 和 y 进行加减乘除操作，并分别保存至变量 a、b、c 和 d 中。

【运行结果】该程序的执行结果如图 3-7 所示。

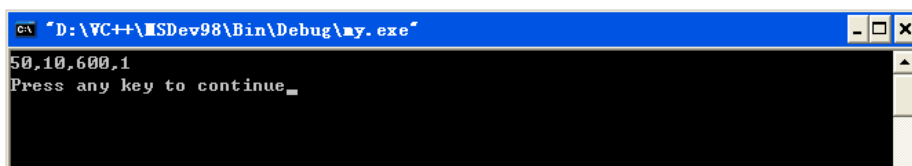


图 3-7 范例 3.7 结果图



注意：每一个类型都有其表示范围，例如 int 型取值范围为-32768~32767，若 x=46423 则其值超过 int 型表示范围，在 C 语言中称为数据溢出。

3.3.3 浮点型

浮点型分为单精度浮点型和双精度浮点型两类。

(1) 单精度浮点型 (float)

字长为 4 字节，占位 32 位，取值范围为 $3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$ 。



(2) 双精度浮点型 (double)

字长为 8 字节，占位 64 位，取值范围为 $1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$ 。

浮点型又可分为浮点型常量和浮点型变量。

1. 浮点型常量

- 小数形式：全部由数字和小数点组成，例如 34.0、89.5 等。
- 指数形式：由数字及其幂指数组成，例如 34.567 可以表示为 0.34567×10^2 、 3.4567×10^1 、 34.567×10^0 、 345.67×10^{-1} 和 3456.7×10^{-2} 。在 C 语言中用 “E” 或 “e” 后加一个整数，表示以 10 为底的幂函数。例如 0.34567×10^2 可以表示为 0.34567E2。



注意：

- (1) 所有浮点型常数默认类型为 double 类型。
- (2) E 前后必须有数字，而且 E 后面的数字必须是整数，E 与数字之间不能有空格。
- (3) 若小数点后位数比较多，应选用 double 类型，可以更精确地表示小数。

2. 浮点型变量

- 单精度浮点型变量：定义为单精度浮点型的变量。定义语句如下：

```
float a,b;
```

- 双精度浮点型变量：定义为双精度浮点型的变量。定义语句如下：

```
double a,b;
```

【范例 3.8】 利用浮点型变量，从键盘输入数据并输出至屏幕。

范例 3.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      float a;                      /*定义一个浮点型变量 a*/
05      scanf("%f",&a);                /*调用 scanf() 函数实现数据的输入*/
06      printf("从键盘输入浮点数为:%f\n",a); /*调用 printf() 函数输出从键盘输入的字符*/
07  }
```

【代码分析】 本例为浮点型简单范例，详细代码分析如下：

- 第 4 行，定义一个浮点型变量 a。
- 第 5 行，调用 scanf() 函数实现浮点型数据的输入。

【运行结果】 该程序的执行结果如图 3-8 所示。



注意： %f 默认精度为输出小数点后 6 位，若想精确到某一位，则可以通过在小数点后加确定的数值。例如 %.2f 表示精确到小数点后两位。

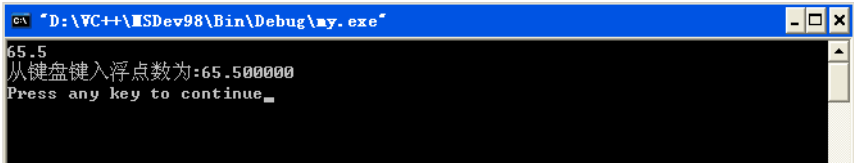


图 3-8 范例 3.8 结果图

3.3.4 字符型

字符型数据有字符型常量和字符型变量之分。

1. 字符型常量

直接用单引号括起来的字符，称为字符型常量。例如，'a'、'l'、'#'和'@'等。

在 C 语言中一般每个字符都对应着一个 ASCII 码值，因此可用 ASCII 码值来表示一个字符。如十进制数 85 可以表示大写字母“U”。

使用字符型常量应注意以下三个方面。

- 大写字母和小写字母的含义不同，例如 ‘c’ 和 ‘C’ 是两个完全不同的字符型常量。
- 字符型常量一定要用单引号括起来，如 a，这样写是不对的。
- 空格也算是字符型常量，若要表示空格，则在单引号中加个空格即可，如 ‘ ’，不可表示为 ‘ ’，中间必须有空格。

C 语言中有以 “\” 开头并以单引号括起来的控制符，有着特定的 ASCII 码值，这种控制符称为转义字符。常用转义字符表如表 3-4 所示。

表 3-4 常见转义字符含义及其 ASCII 码值

转义字符	含 义	ASCII 码值
\n	换行符，换行，光标移至下一行	10
\0	输出空值	0
\r	回车符，光标移到本行开头	13
\b	退格符，光标往前退一格	8
\f	换页符，光标移到下一页的开始	12
\t	水平制表符，光标水平移动一个 Tab	9
\v	垂直制表符，光标垂直移动一个 Tab	11
\a	响铃	7
\\	输出反斜杠 “\”	92
\'	输出单引号 “'”	39
\"	输出双引号 “”	34
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

使用转义字符时应注意：

- 转义字符常量只能代表一个字符，如\n'，\140'都只代表一个字符。
- 反斜杠后的十六进制数要用小写字母开头，如\x0c'（或\x0C'）表示\f'。



- 转义字符一般用在 `printf()` 函数中，在 `scanf()` 函数一般不使用。如 `printf("%d\n")` 表示输出一个数据后换行。

2. 字符型变量

字符型变量可以用来存储字符型常量，一个字符型变量只能存储一个字符型常量，不能存放多个字符型常量。

字符型变量定义语句如下：

```
char c;
```

字符型变量根据修饰符的不同，可分为有符号和无符号的字符型变量。字符在机器中占 1 个字节，以 ASCII 码值表示。有符号字符变量的取值范围为 -128~127，无符号字符变量取值范围为 0~255。

把一个字符型常量放置字符型变量中，实际上并不是把字符型常量放置变量对应的地址中，而是将该字符常量对应的 ASCII 码值转化为二进制数存储至变量地址中。例如要将 'a' 存放至字符变量 `c` 中，字符 `a` 对应的 ASCII 码值为 97，二进制数为 0110001，实际上是将 0110001 存储至变量 `c` 对应的地址中。要读取变量 `c` 中的字符，则将二进制数转化为相应的字符输出即可。

【范例 3.9】 对字符型数据进行算术运算，输出其运算结果。

分析：定义字符型变量用于保存要计算的字符型数及计算得出的结果，再利用 `printf()` 函数输出结果即可。

范例 3.9 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      char c1='A',c2=96,c;           /*定义 3 个字符型变量*/
05      c=c2-c1;                       /*将 c2 减去 c1 的差值赋给变量 c*/
06      printf("%d,%c\n",c,c);        /*输出变量 c 对应的数值及字符*/
07  }
```

【代码分析】 本例是字符型的简单应用，详细代码分析如下：

- 第 4 行，定义了三个字符型变量，`c1` 赋值为 'A'，`c2` 赋值 96，即 ASCII 码值为 96，对应字母 'a'。
- 第 5 行，变量 `c` 保存 `c2` 减去 `c1` 的值，字符型常量相减是用其 ASCII 码值相减的。

【运行结果】 该程序的执行结果如图 3-9 所示。

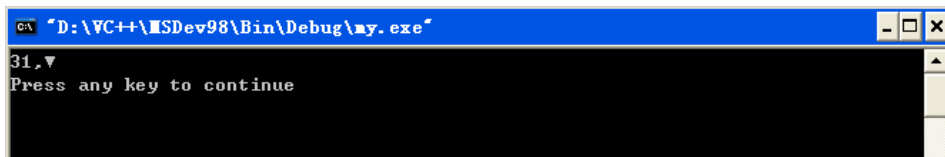



图 3-9 范例 3.9 结果图



3.4 数据机内存储形式

C 语言中整型、浮点型、字符型都是程序中的基本数据类型，了解这些类型在计算机内存中存储的形式对程序设计中类型的选取及使用有着很大的帮助。本节将介绍这些数据类型在计算机内存中具体的存储形式。

3.4.1 整型数据机内存储形式

计算机数据取值范围与其所占的位数有关，数据占的位数越多，其数据范围也就越大。就好比两位数和三位数，两位数取值范围为 0~99，三位数的取值范围为 0~999，很明显三位数的取值范围要比两位数的范围大。

计算机中每种数据类型占的位数不同，因此其取值范围也是不同的。在 C 语言中，整型数据占 16 位。在计算机中，每 8 位称为 1 个字节，因此 C 语言中的整型数据也可以说成占 2 个字节。计算机数据采用二进制表示，由一串 0 和 1 组成的数字表示。

根据数据有无符号，C 语言整型数据可分为 16 位有符号整型和 16 位无符号整型，其取值范围如表 3-5 所示。

表 3-5 整型类型及其范围

整型类型	位 数	数据的取值范围
16 位有符号整型	16	-32768~32767 ($-2^{15} \sim 2^{15}-1$)
16 位无符号整型	16	0~65535 ($0 \sim 2^{16}-1$)

因此，在程序设计的过程中，应根据实际情况不同选用合适的数据类型。

3.4.2 浮点型数据机内存储形式

C 语言中提供了两种浮点型：单精度（float）和双精度浮点型（double）。单精度浮点型占 32 位，即 4 个字节，双精度浮点型占 64 位，即 8 个字节。

浮点型数据在计算机中存储分为尾数和指数两个部分。尾数部分表示数的有效数字，尾数部分位数越多，表示的数就能越精确。指数部分决定数据的取值范围，指数部分位数越多，数的取值范围就越大。

单精度和双精度浮点型取值范围也不同，具体如表 3-6 所示。

表 3-6 单精度和双精度浮点型有效位数及其范围

浮点类型	有效数字	取值范围
单精度（float）	7 位	$10^{-38} \sim 10^{38}$
双精度（double）	15~16 位	$10^{-308} \sim 10^{308}$

3.4.3 字符型数据机内存储形式

字符包括字母（F、J、a）和特殊字符（*、&、#）。C 语言中每一个字符有着一个对应的



ASCII (American Standard Code for Information Interchange, 美国信息互换标准代码) 码值, 即对应的二进制码值。ASCII 是以拉丁字母为基础的一套编码系统。

ASCII 码用 8 位二进制数来表示一个字符, 可以表示 128 种字符。例如 ‘a’ 对应的 ASCII 码值为 97, 用二进制表示为 011000001, 存储方式如下:

011000001 (‘A’)

字符型数据可以与整型、浮点型数据进行算术运算, 如 $(\text{'a'}+5)*3.0$ 。不同数据类型之间进行运算时, 要先把不同的类型转换为相同的类型再进行运算。short 型一般转化为 int 型, float 型转化为 double 型, 以提高运算的精准度。例如 float 型和 double 类型的数据相加时, float 类型的数据要先转化为 double 类型, 再与 double 类型数据相加, 最后输出 double 类型结果。



3.5 局部变量

局部变量是指在函数内部定义的变量。局部变量有一个很重要的特性, 即当且仅当该函数被执行时, 局部变量才会产生, 当函数程序结束时局部变量则会消失。例如下面两个函数:

```
void f1()
{
    int a;
    a=20;
}
void f2()
{
    int a;
    a=10;
}
```

在上面的两个函数 f1() 和 f2() 中, 整型变量 a 被说明了两次, 但是不会冲突。因为 f1() 和 f2() 函数中的 a 变量是局部变量, 只在函数内有效。



说明: C 语言中有一个关键字 auto, 但一般不用, 因为当 auto 省略时, 非全局变量都会默认为局部变量。

局部变量只有在执行特定的语句时, 才会产生, 一般是在函数或复合语句中。当函数或复合语句执行完后, 局部变量也会消失。这样可以实现在需要它时才给其分配空间, 在很大程度上可以提高内存的存储空间的利用率。



3.6 全局变量

全局变量与局部变量不同, 它在程序的整个运行过程中都存在, 可以被任何一段程序调用。全局变量要声明在函数的外面, 可以被整个程序中的代码随意使用。



全局变量除了要说明在其调用之前，并且在函数之外，可以说明在任何位置。一般全局变量都是定义在程序的开头，可以一目了然，如下所示。

```
int score;
void main()
{
    score=90;
    f1();
}
void f1()
{
    int s;
    s=score/10;
    printf("score is %d\n",score);
}
void f2()
{
    int score=77;
    printf("score is %d\n",score);
}
```

在上面的程序中，程序开头定义了一个 `score` 全局变量，`main()`和 `f1()`函数中都没有说明 `score` 变量，但可以使用该变量。在 `f2()`函数中定义了一个名为 `score` 的局部变量，当在 `f2()`中调用 `score` 时使用的是局部变量 `score`，而不是全局变量 `score`，因为局部变量会把全局变量覆盖，但其值不会影响全局变量。

全局变量在内存有特定的存储空间，当程序中有多个函数要使用某一变量时，则将其定义为全局变量很方便。全局变量也有很多的缺点，主要包含以下两个方面：

- (1) 全局变量在程序的整个运行过程中都占据着内存，导致存储空间利用率不高。
- (2) 全局变量很容易受到程序的一些代码影响而发生改变，导致程序运行出错。



3.7 形式参数

形式参数是指用来接受函数要使用的参数的变量，简称为形参。形参可以像局部变量一样在函数内部使用，它们说明在函数名后，花括号之前，如下所示。

```
void f(a,b)
char *a;
char b;
{
    if(*a!=b)
        printf("error\n");
    else
        printf("right\n");
}
```



上面的程序中，有两个形参，分别是 `a` 和 `b`。上述形参经过说明之后，在函数中便可像局部变量一样使用。它与局部变量一样，函数运行时产生，退出时消失。

形式参数可以把外部参数传给函数内部。在调用时，实参应该保证与函数的形参类型一致，否则程序会出错，详细情况将在后面函数的章节中讲解到。



3.8 赋值及类型转换

赋值是指把一个表达式赋给一个变量，形式如下：

变量=表达式；

上式中表达式可以是一个常量，也可以是常量与变量组合的表达式。C 语言中用“=”表示赋值操作，等号的左边必须是一个变量，右边可以为任意的表达式。全局变量没有初始化时，系统默认值为零。

赋值类型转换指的是赋值语句中等号两边数据类型不同，将其转换为同一类型的过程。它遵循等号右边的类型转换为等号左边类型的规则，如下例所示。

```
int a;
float b;
char c;
void f()
{
    c=a;
    a=b;
    b=c;
}
```

在上面的程序中，字符型变量 `c` 被赋予整型变量 `a`，字符型占 8 位，整型占 16 位，所以在赋值的过程中是把整型变量 `a` 的低 8 位赋给 `c`，高 8 位省去。“`a=b`”语句把浮点型数据赋给整型变量，即把 `b` 的整数部分（非小数数字）赋给 `a`。“`b=c`”语句，字符型数据赋给浮点型数据，即把 `c` 变量对应的整数值转化为浮点数赋给 `b`。

在使用赋值类型转换时，精度低的类型转换为精度高的类型不会有很大的影响，如 `int` 型转化为 `float` 型或 `double` 型，精度会更高。精度高的类型转换为精度低的类型则会造成数据的丢失，例如 `float` 型转化为 `int` 型，小数点后的数字会丢失，编写程序应注意此问题。



3.9 运算符及其表达式

在现代计算机中包含 `+`、`-`、`*`、`/` 等运算符，而在 C 语言中除了这些运算符还有很多其他的运算符，本节将会重点讲解这些运算符的概念。

C 语言中的运算符归纳起来总共有算术运算符、赋值运算符、关系运算符、逻辑运算符、位运算符、条件运算符、逗号运算符、指针运算符、求字节运算符、强制类型转换运算符、分



量运算符、下标运算符和其他运算符 13 种。根据运算符的目，又可以分为单目运算符、双目运算符和三目运算符 3 种，如下所示。

- 单目运算符：只有一个对象，如++、*等。
- 双目运算符：表示有两个对象，如+、-、*等。
- 三目运算符：表示有三个对象。C 语言中只有一个条件运算符是三目运算符，在后面将会讲解到。

3.9.1 算术运算符及其表达式

算术运算符主要包括+、-、*、/、%等运算符，这些运算符都是双目运算符，即连接两个对象。算术运算符中*、/、%优先级比+、-高，*、/、%同一级别，+、-同一级别。运算方向是从左至右的。算术运算表达式是指用算术运算符把对象连接起来的符合 C 语言语法的式子。

【范例 3.10】编写一个程序，利用算术运算符实现简单的运算。

范例 3.10 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=5,y=6;           /*定义两个整型变量 x 和 y，初始化其值为 5 和 6*/
05      int a=x-y;             /*将 x 减去 y 的值赋值给变量 a*/
06      int b=x+y;             /*将 x 加上 y 的值赋值给变量 b*/
07      int c=x*y;             /*将 x 与 y 的乘积赋值给变量 c*/
08      int d=x/y;             /*将 x 除以 y 的值赋值给变量 d*/
09      printf("%d,%d,%d,%d\n",a,b,c,d); /*调用 printf()函数输出计算得出的结果*/
10  }
```

【代码分析】本例为算法运算符的简单应用，详细代码分析如下：

- 第 4 行，定义了两个变量 x 和 y，初始化其值分别为 5 和 6。
- 第 5~8 行，对 x 和 y 分别进行+、-、*、/操作并赋给变量 a、b、c、d。

【运行结果】该程序的执行结果如图 3-10 所示。

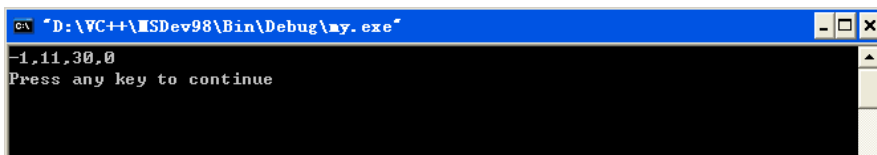


图 3-10 范例 3.10 结果图

3.9.2 加 1 和减 1 运算符

C 语言中有着两个很特殊的运算符，它们分别是加 1 和减 1 运算符，符号为++和--。++运算符是将其操作数加 1，--运算符则与++相反，将其操作数减 1。例如：

```
a++;
a--;
```



由浅入深学 C 语言——基础、进阶与必做 430 题

上面程序中 `a++` 等价于 `a=a+1`；即 `a` 加上 1 再赋给 `a`，`a--` 等价于 `a=a-1`；即 `a` 减去 1 赋给 `a`。

加 1 和减 1 运算符可以放在操作数的前后，虽然单独写没什么区别，但在表达式中有很大的区别。例如：

```
a=10;
a++;
++a;
a=10;
y=a++;
a=10;
y=++a;
```

`a++` 和 `++a` 单独运行时，都表示 `a=a+1`，即加 1 操作。但在表达式中，若运算符在操作数之后，表示先引用该变量再进行加 1 或减 1 操作；若运算符在操作数之前，则先对操作数进行加 1 或减 1 操作，再引用该变量。如上面程序中，`y=a++`；直接引用变量 `a`，`y` 的结果就为 10，`y=++a`；则先将 `a` 进行加 1 操作，再将变量 `a` 赋给 `y`，`y` 的值为 11。

【范例 3.11】 编写一个程序，利用加 1 和减 1 运算符实现简单的运算。

范例 3.11 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=3,y=5;                /*定义两个整型变量 x 和 y，初始化其值为 3 和 5*/
05      int a=x++;                  /*将 x++ 的值赋给变量 a*/
06      int b=x--;                  /*将 x-- 的值赋给变量 b*/
07      int c=++y;                  /*将 ++y 的值赋给变量 c*/
08      int d=--y;                  /*将 --y 的值赋给变量 d*/
09      printf("%d,%d,%d,%d\n",a,b,c,d); /*调用 printf() 函数输出 a, b, c, d 的值*/
10  }
```

【代码分析】 本例为加 1 和减 1 运算符简单应用，详细代码分析如下：

- 第 5 行，将 `x++` 赋值给变量 `a`，因为 `++` 运算符在变量之后，因此其值为 `x` 的值。
- 第 6 行，将 `x--` 赋值给变量 `b`，因为 `--` 运算符在变量之后，因此其值为 `x` 的值。
- 第 7 行，将 `++y` 赋值给变量 `c`，因为 `++` 运算符在变量之前，因此其值为 `y` 的值加 1。
- 第 8 行，将 `--y` 赋值给变量 `d`，因为 `--` 运算符在变量之前，因此其值为 `y` 的值减 1。

【运行结果】 该程序的执行结果如图 3-11 所示。

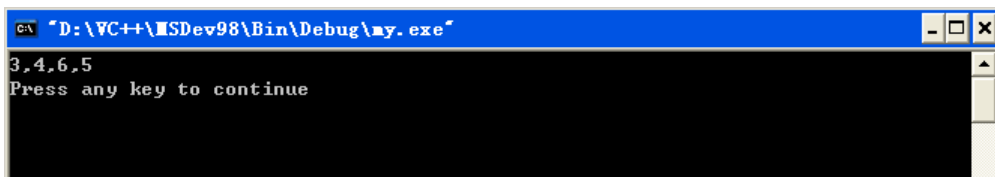


图 3-11 范例 3.11 结果图

3.9.3 关系运算符及其表达式


关系运算符包括 6 种，分别是：

<	<=	>	>=	==	!=
小于	小于或等于	大于	大于或等于	等于	不等于

算术运算符的优先级比关系运算符的优先级高，如：

```
a+b>c+d    a-b<d
```

关系运算符主要通过运算符进行两边的比较，通常运用在 if 等语句中作为条件。



注意：==和=号的区别，==是判断两个数是否相等，=是把右边的表达式赋值给左边的变量，初学时很容易搞混。

关系表达式是指用关系运算符把对象连接起来的符合 C 语言语法的式子。

【范例 3.12】编写一个程序，通过关系运算符进行简单的运算。

分析：关系运算符包括小于、小于或等于、大于、大于或等于、等于、不等于 6 种运算符，不同的运算符可以用来对数据进行不同的比较。

范例 3.12 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=5,y=7;                /*定义两个整型变量 x 和 y*/
05      int a=(x<y);                /*比较 x 是否小于 y，将其真值赋给变量 a*/
06      int b=(x!=y);              /*比较 x 是否不等于 y，将其真值赋给变量 b*/
07      int c=(x==y);              /*比较 x 是否等于 y，将其真值赋给变量 c*/
08      int d=(x>y);                /*比较 x 是否大于 y，将其真值赋给变量 d*/
09      printf("%d,%d,%d,%d\n",a,b,c,d);
10  }
```

【代码分析】本例为关系运算符应用范例，详细代码分析如下：

- 第 5 行，判断变量 x 是否小于 y，并将其逻辑值赋给变量 a。
- 第 6 行，判断变量 x 是否不等于 y，并将其逻辑值赋给变量 b。
- 第 7 行，判断变量 x 是否等于 y，并将其逻辑值赋给变量 c。
- 第 8 行，判断变量 x 是否大于 y，并将其逻辑值赋给变量 d。

【运行结果】该程序的执行结果如图 3-12 所示。

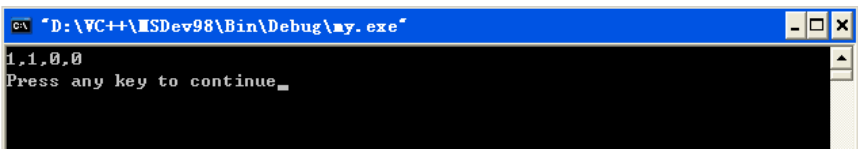


图 3-12 范例 3.12 结果图



3.9.4 逻辑运算符及其表达式

逻辑运算符包括三种，分别是!、&&、||，分别代表非、与、或。逻辑表达式是指用逻辑运算符把对象连接起来的符合 C 语言语法的式子。逻辑表达式的运算结果只能为真和假，在 C 语言中分别用 1 和 0 表示。其运算规则如表 3-7 所示。

表 3-7 运算规则表

a	b	!a	!b	a&&b	a b
真 (1)	真 (1)	假 (0)	假 (0)	真 (1)	真 (1)
真 (1)	假 (0)	假 (0)	真 (1)	假 (0)	真 (1)
假 (0)	真 (1)	真 (1)	假 (0)	假 (0)	真 (1)
假 (0)	假 (0)	真 (1)	真 (1)	假 (0)	假 (0)

若&&运算符组成一个逻辑表达式，如下：

(表达式 1)&&(表达式 2)&&...

只要其中一个表达式的逻辑值为 0，则不用计算后面表达式的值，该式子的值就为 0。

若||运算符组成一个逻辑表达式，如下：

(表达式 1)|| (表达式 2) ||...

只要其中一个表达式的逻辑值为 1，则不用计算后面表达式的值，该式子的值就为 1。

若是一个复杂的逻辑表达式，则按照从左至右的原则进行计算。

3.9.5 三目运算符

三目运算符的表示一般为“?:”，该运算符连接三个对象，是 C 语言中唯一的一个三目运算符，又称条件运算符。它的一般形式如下：

表达式 a?表达式 b: 表达式 c

其执行步骤如下：

- (1) 计算表达式 a 的值。
- (2) 如果表达式 a 的值为 1，则执行表达式 b。
- (3) 如果表达式 b 的值为 0，则执行表达式 c。

有多个三目运算符时，按从右至左的规则运算。例如下面两个表达式是等价的：

```
a<b?b:c>b?c:b;
a<b?b:(c>b?c:b);
```

【范例 3.13】编写程序，实现三目运算符的简单应用。

分析：三目运算符判断条件表达式的真值，若为真则执行?号后的第一个表达式，否则执行第二个表达式。

范例 3.13 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=5;                /*定义一个整型变量 x*/
05      int y=++x>5?0:1;        /*通过三目运算符对变量 x 进行运算*/
06      printf("%d,%d\n",x,y);  /*调用 printf() 函数输出变量 x 和 y 的值*/
07  }
```

【代码分析】本例为三目运算符应用范例，详细代码分析如下：

- 第 5 行，判断++x 的值是否大于 5，若大于 5 赋值 0 给变量 y，否则赋值 1 给变量 y。

【运行结果】该程序的执行结果如图 3-13 所示。

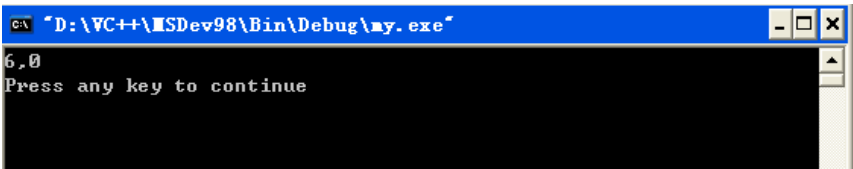


图 3-13 范例 3.13 结果图

3.9.6 位运算符

C 语言有着低级语言的一些特性，它可以按位运算实现对计算机硬件的直接操作。位运算符是一种对计算机中二进制数操作的运算符，包括按位与运算、按位或运算、按位求反运算、按位异或运算、左移运算、右移运算 6 种，具体情况如表 3-8 所示。

表 3-8 位运算种类及其作用

位运算符	功 能	运算规则
~	按位逻辑反	从右至左
<<	按位左移	从左至右
>>	按位右移	从左至右
&	按位逻辑与	从左至右
^	按位逻辑异或	从左至右
	按位逻辑或	从左至右

(1) 按位与运算。按位运算符&可用来使某些位置 0，两个操作数中对应的位只要有一个为 0，则该位结果就为 0。结合规则如下：

```
0&0=0  0&1=0  1&0=0  1&1=1
```

例如：两个二进制数 10000011 和 00001101 进行按位与结果为多少？

10000011 和 00001101 进行按位与运算，按其规则，只要有一位为 0 则为 0，得出两者运算的结果为 00000001。

(2) 按位或运算。按位运算符|与按位与运算相反，用来使某些位置 1，两个操作数中对应的位只要有一个为 1，则该位结果就为 1。结合规则如下：



$0|0=0$ $0|1=1$ $1|0=1$ $1|1=1$

例如：两个二进制数 10000011 和 00001101 进行按位或结果为多少？

10000011 和 00001101 进行按位或运算，按其规则，只要有一位为 1 则为 1，得出两者运算的结果为 10001111。

(3) 按位异或运算。按位异或运算是将两个数每位进行比较，若不同则为 1，否则为 0。其结合规则如下：

$0^0=0$ $0^1=1$ $1^0=1$ $1^1=0$

例如：两个二进制数 10000011 和 00001101 进行按位异或结果为多少？

10000011 和 00001101 进行按位异或运算，按其规则，若对应位数相同则为 0，否则为 1，得出两者运算的结果为 10001110。

(4) 按位非运算～。按位非运算用来求一个二进制数的反码，即把操作数每一位中的 1 变为 0，0 变为 1。

例如：二进制数 10000011 进行按位非运算结果为多少？

10000011 进行按位非运算，很容易得出其结果为 01111100。

(5) 左移运算。左移运算符<<是将二进制数的每一位往左移动若干位，将高位省略，右边的低位则补 0。

例如：10000011<<2 的结果为多少？

10000011<<2，根据其定义运算得 00001100。

(6) 右移运算。右移运算符>>是将二进制数的每一位往右移动若干位，将低位省略，左边的高位补 0。

例如：10000111>>2 的结果为多少？

10000111>>2，根据其定义运算得 00100001。

【范例 3.14】编写一个程序，对数据进行简单的位运算。

分析：位运算符包含左移、右移、取非、与、或、异或 6 种运算符，分别可以用来实现不同的功能。

范例 3.14 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=5,y=8;                /*定义两个变量 x 和 y，初始值为 5 和 8*/
05      int a=x>>1;                /*将变量 x 值右移一位赋值给变量 a*/
06      int b=y<<1;                /*将变量 y 的值左移一位赋值给变量 b*/
07      int c=~x;                  /*将变量 x 按位取反赋值给变量 c*/
08      int d=x^y;                 /*将变量 x 异或 y 的值赋给变量 d*/
09      printf("%d,%d,%d,%d\n",a,b,c,d);
10  }
```

【代码分析】本例为位运算符应用范例，详细代码分析如下：



- 第5行，将变量 x 右移一位，并将其值赋给变量 a。
- 第6行，将变量 y 左移一位，并将其值赋给变量 b。
- 第7行，对变量 x 进行按位取反运算，再将其值赋给变量 c。
- 第8行，对变量 x 和 y 进行按位异或运算，赋值给变量 d。

【运行结果】该程序的执行结果如图 3-14 所示。

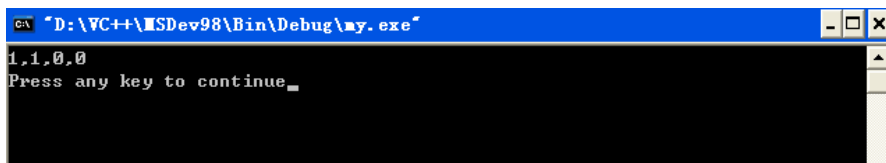


图 3-14 范例 3.14 结果图

3.9.7 sizeof 运算符

sizeof 运算符是 C 语言提供的一个用于返回变量或类型修饰符字节长度的运算符。sizeof 的一般形式为：

```
sizeof(名称);
```

例如：

```
void main()
{
    int a;
    a=10;
    printf("%d,%d\n",sizeof(a),sizeof(int));
}
```

不同类型的数据在计算机中占的字节不同，同一种类型的数据在不同的系统中也会有所差异。sizeof 运算符可以确定数据类型所占字节的位数，从而可使程序在多个系统中运行，提高程序的可移植性。

【范例 3.15】编写程序，通过 sizeof 运算符计算变量存储空间大小。

分析：sizeof 为 C 语言提供的运算符，可以用来计算任意变量在系统中的存储空间大小。

范例 3.15 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int a[5];
05      printf("%d\n",sizeof(a));
06  }
```

【代码分析】本例为 sizeof 运算符的简单应用，详细代码分析如下：

- 第4行，定义了一个整型变量数组 x。



- 第 5 行，利用 `sizeof` 运算符计算数组 `x` 存储空间大小并输出结果至屏幕。
- 【运行结果】该程序的执行结果如图 3-15 所示。

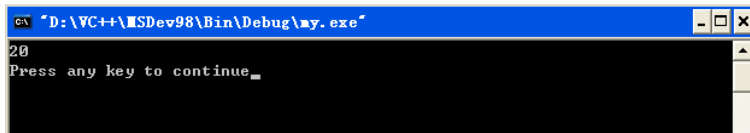


图 3-15 范例 3.15 结果图

3.9.8 逗号运算符

逗号运算符可以把多个表达式组合在一起，逗号运算符返回的值为逗号右边的值。如：

```
y=(x=1,x+2);
```

首先将 1 赋给 `x`，再做加 2 操作，将逗号右边的值赋给 `y`，即 `y` 的值为 3。

逗号运算符本质上是多个运算组合在一起，从左至右逐个运算，最后将逗号右边的值赋值给等号左边的变量。例如：

```
x=1;  
y=(x=x+2,x=x*3,x-5);
```

首先执行逗号最左边的表达式 `x=x+2`，`x` 的值变为 3，其次执行 `x=x*3`，`x` 值为 9，最后执行 `x-5` 操作，`y` 的值为 4。

【范例 3.16】根据给出的数据，对其进行算术运算和逻辑运算。

范例 3.16 代码实现

```
01  #include <stdio.h>  
02  void main()  
03  {  
04      int x=0,y=1,z;  
05      float a=5.7;  
06      char b='z';  
07      z=(++x)&&(--y);  
08      printf("%d\n",z);  
09      z=a-y;  
10      printf("%d\n",z);  
11      z=b;  
12      printf("%d\n",z);  
13  }
```

【代码分析】本例为一个数据运算的范例，详细代码分析如下：

- 第 4~6 行，定义了要进行操作的变量，并对一些变量进行了初始化。
- 第 7 行，将 `x` 加 1，`y` 减 1 再进行与操作赋给变量 `z`。
- 第 9 行，将变量 `a` 减去 `y` 的值赋给变量 `z`。



- 第11行，将字符型变量b赋给z，即b变量存储字符对应的ASCII码值赋给z。

【运行结果】该程序的执行结果如图3-16所示。

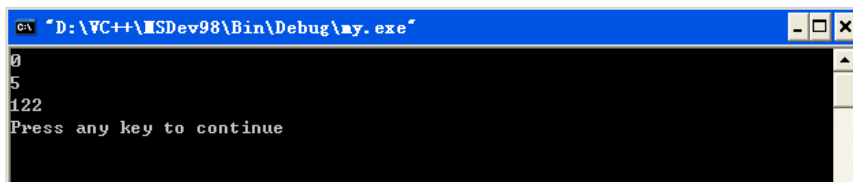


图3-16 范例3.16结果图



注意：程序运行至第9行 $z=a-y$ ；，此时y的值已经改变。在第7行中y的值已经减1，变为0，所以z的值为5，而不是4。



3.10 小结

本章讲述了C语言中数据的基本类型、常量、变量和运算符等知识，主要包括以下三方面的内容：

- (1) C语言中包含了丰富的数据类型，如整型、浮点型、字符型等，其中每一种数据类型占据的存储空间大小及格式都是不一样的。
- (2) C语言中包含着丰富的运算符，如关系运算符、比较运算符、逗号运算符等。
- (3) 每一种运算符都有着不同的优先级别，在使用过程中应特别注意。



3.11 习题

一、选择题

1. 以下程序的运行结果为 ()。

```
void main()
{
    int a=1,b=2,c;
    c=(a-1)&&(b--);
    printf("%d,%d",b,c);
}
```

A. 2, 1

B. 2, 0

C. 1, 1

D. 1, 0

【提示】变量a的值为1，变量b的值为2，a-1的值为0。因此在计算 $(a-1)\&\&(b--)$ 时，b--操作忽略不计。

2. 以下程序运行结果正确的是 ()。

```
void main()
{
```



```
int x=3,y=4,z,s;  
z=y||x--;  
s=x^y;  
printf("%d,%d\n",x,s);  
}
```

- A. 3, 6 B. 3, 7 C. 2, 6 D. 2, 7

【提示】表达式 $y||x--$ 中， y 的值为 4，逻辑值为真，因此 $x--$ 操作被忽略，即不进行运算。变量 s 中保存变量 x 和 y 异或的结果，调用 `printf()` 函数输出变量 x 和 s 的值。

3. 以下程序的运行结果为 ()。

```
void main()  
{  
    int a,b,x,y;  
    a=5;  
    b=6;  
    x=++a;  
    y=b++;  
    printf("%d,%d",x,y);  
}
```

- A. 5, 6 B. 6, 6 C. 6, 5 D. 5, 5

【提示】初始化变量 a 和 b 的值为 5 和 6，将 $++a$ 的值赋值给变量 x ，即 a 的值加 1，将 $b++$ 的值赋值给变量 y ，即变量 b 的值，调用 `printf()` 函数输出变量 x 和 y 的值。

二、填空题

1. 设 $t=10$ ，写出下列表达式的运算结果。

- (1) $t++$ 的值为_____。
- (2) $t--$ 的值为_____。
- (3) $t+=t$ 的值为_____。
- (4) $t\%=(t\%3)$ 的值为_____。
- (5) $t*=t$ 的值为_____。

【提示】 t 的初始值为 10，因此 $t++$ 的值为 10， $t--$ 的值为 10， $t+=t$ 的值为 20， $t\%=(t\%3)$ 的值为 0， $t*=t$ 的值为 100。

2. 下列程序的运行结果为_____。

```
void main()  
{  
    float x=1.3,y=13.6e-1,z,t;  
    z=x+y;  
    t=(int)x;  
    printf("z=%d,t=%d",z,t);  
}
```

【提示】将变量 x 和 y 的和赋值给变量 z ，将变量 x 转化为 `int` 型赋值给变量 t ，然后输出变量 z 和 t 的值。



3. 以下程序的运行结果为_____。

```
void main()
{
    int x,y,a,b;
    a=30;
    b=40;
    x=a+4;
    y=b--;
    printf("%d,%d,%d,%d\n",x,y,a,b);
}
```

【提示】将变量 `a+4` 的值赋给变量 `x`，`b--` 值赋给变量 `y`。

4. 以下程序的运行结果为_____。

```
void main()
{
    char x='a',y='b';
    printf("%d,%d\n",x,y);
}
```

【提示】将字符 `a` 和字符 `b` 分别赋给变量 `x` 和 `y`，调用 `printf()` 函数输出字符变量 `x` 和 `y` 对应的 ASCII 码值。

5. 以下程序的运行结果为_____。

```
void main()
{
    int a=4,b=5;
    a=4+3;
    printf("%d",a);
    b=a;
    printf("%d",b);
}
```

【提示】定义变量 `a` 和 `b`，初始化其值为 4 和 5。将 `4+3` 的值即 7 赋值给变量 `a`，输出变量 `a` 的值，再将变量 `a` 的值赋值给变量 `b`，输出变量 `b` 的值。

6. 以下程序运行时，输入 32 45，则其输出结果为_____。

```
void main()
{
    int x,y;
    scanf("%d%d",&x,&y);
    printf("%d,%d",x,y);
}
```

【提示】从键盘键入 32 45，则 `x` 和 `y` 的值分别为 32 和 45，输出变量 `x` 和 `y` 的值至屏幕。

7. 以下程序运行结果为_____。



```
void main()
{
    int x,y;
    x=5;
    y=2*x+4;
    printf("%d,%d\n",x,y);
}
```

【提示】定义变量 x 和 y ，将 5 赋值给变量 x ，将 $2*x+4$ 的值赋值给变量 y ，输出变量 x 和 y 的值。

8. 以下程序运行结果为_____。

```
void main()
{
    int a=5,b=6,t;
    a=b;t=a;b=a;
    printf("%d,%d",a,b);
}
```

【提示】定义两个变量 a 和 b ，其初始值为 5 和 6，通过变量 t 交换变量 a 和 b 的值，再输出变量 a 和 b 的值。

9. 以下程序运行结果为_____。

```
void main()
{
    int a=5,b=6;
    int i=3,j=4;
    i=a++;
    j=--b;
    printf("%d,%d,%d,%d\n",a,b,i,j);
}
```

【提示】将 $a++$ 的值赋给变量 i ， $--b$ 的值赋值给变量 j ，输出变量 a ， b ， i ， j 的值。

10. 以下程序运行结果为_____。

```
void main()
{
    int a=013,b=010;
    printf("%d,%d",a,b);
}
```

【提示】变量 a 中保存的数为八进制数 13，变量 b 中保存的数为八进制数 10，通过 `printf()` 函数将其转换为十进制输出。

11. 以下程序运行结果为_____。

```
void main()
{
    char c='c'+55-'a';
}
```



```
printf("%d,%c",c,c);
}
```

【提示】将'c'+55-'a'赋值给字符变量 c，输出字符变量中的字符以及对应的 ASCII 码值。

12. 以下程序运行结果为_____。

```
void main()
{
    int x=1,y=2,t;
    char a='b',b='g',c,d;
    t=x;
    x=y;
    y=t;
    c=++a;
    d=b--;
    printf("%d,%d,%c,%c",x,y,c,d);
}
```

【提示】通过变量 t，交换变量 x 和 y 的值，将++a 的值赋给变量 c，b--的值赋给变量 d，最后输出 x、y、c、d 的值。

三、编程题

1. 从键盘输入 3 个数，判断其是否能构成直角三角形，若能则输出该直角三角形的面积，否则输出不能构成三角形的提示信息。

【提示】定义三个变量保存三边，判断三边大小是否满足构成直角三角形的条件，若能构成三角形则计算其面积，若不能构成三角形则提示相关的信息。

【核心代码】

```
float a,b,c;
if(a*a+b*b==c*c)
    s=0.5*a*b;
if(c*c+b*b==a*a)
    s=0.5*c*b;
if(a*a+c*c==b*b)
    s=0.5*a*c;
.....
```

2. 编写一个程序，输入一个华氏温度，按照公式 $C=5/9*(F-32)$ 输出摄氏温度。

【提示】定义一个浮点型变量保存华氏温度，按照公式计算其摄氏温度输出即可。

【核心代码】

```
float F,C;
scanf("%f",&F);
C=5/9*(F-32);
printf("%f\n",C);
```

3. 输入一个年份，判断该年是否为闰年，若为闰年则输出“yes”，否则输出“no”。



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】若一年可以被 400 整除或能被 4 整除但不能被 100 整除则为闰年，根据条件进行判断并输出即可。

【核心代码】

```
int year;
if(year%400==0||(year%4==0&&year%100!=0))
printf("yes\n");
else
...
```

4. 从键盘输入一个学生的 5 门科目成绩，计算其平均值并输出。

【提示】定义 5 个变量保存成绩，另外定义一个变量保存平均分，计算后输出至屏幕。

【核心代码】

```
float s1,s2,s3,s4,s5;
scanf("%f%f%f%f%f",&s1,&s2,&s3,&s4,&s5);
s=(s1+s2+s3+s4+s5)/5;
printf("the average is %f\n",s);
```

5. 从键盘输入 3 个数，输出其中的最大者。

【提示】输出 3 个数中的最大者，可以对 3 个数两两进行比较得出结果。

【核心代码】

```
scanf("%d%d%d",&x,&y,&z);
if(x>y&&x>z)
max=x;
else if(y>x&&y>z)
max=y;
else
max=z;
```

6. 从键盘输入 3 个数，将其从小到大排序后输出。

【提示】从键盘输入 3 个数，可以通过 scanf() 函数实现。对这 3 个数从小到大排序，可以通过对其进行比较来实现。

【核心代码】

```
for(i=0;i<3;i++)
scanf("%d",&a[i]);
for(i=0;i<2;i++)
for(j=0;j<3;j++)
if(a[i]>a[j])
{
t=a[i];
a[i]=a[j];
a[j]=t;
}
```



7. 编写程序，输出 1~1000 之间的偶数。

【提示】设置一个变量，其值从 1 变化至 1000，对其进行判断再输出。若该数除以 2 的余数为 0 则为偶数，输出至屏幕。

【核心代码】

```
for(i=1;i<=1000;i++)
    if(i%2==0)
        printf("%d ",i);
```

8. 从键盘输入一串字符，输出至屏幕。

【提示】从键盘输入一串字符，可以将保存至字符数组中，然后调用 printf() 函数输出字符数组中的内容至屏幕。

【核心代码】

```
scanf("%s",s);
printf("%s",s);
```

9. 编写一个程序，输入半径，求圆的面积和体积。

【提示】从键盘输入半径，将其保存至变量 r 中。然后套用面积计算公式，计算圆的面积后输出至屏幕。

【核心代码】

```
scanf("%d",&r);
s=3.14*r*r;
printf("圆的面积为%d\n",s);
```

10. 从键盘输入两个数，对其进行加减乘除操作并将结果输出至屏幕。

【提示】调用 scanf() 函数从键盘获取两个数，保存至相应变量中，然后对其进行加减乘除操作输出结果至屏幕。

【核心代码】

```
scanf("%d%d",&x,&y);
a=x+y;
b=x-y;
c=x*y;
d=x/y;
```

第 4 章 顺序结构程序设计

结构化程序设计的概念最早由 E.W.Dijkstra 在 1965 年提出，其主要思想是使用三种程序控制语句来构建程序，这样任何一种算法都可以用顺序、选择、循环三种基本控制语句构建。若算法是程序的灵魂，则结构化程序设计是构成程序的血肉。本书的第 4~6 章将专门介绍结构化程序设计，学习了这三章的内容后，就掌握了 C 语言的基本语法，在实现算法时，就能游刃有余。

本章主要涉及的知识点有：

- 结构化程序设计思想；
- 顺序结构程序设计；
- 输入、输出函数的使用；
- 通配符；
- 取地址符；
- 转义字符；
- 顺序结构程序设计应用。



4.1 顺序结构程序设计初探

在三种基本控制结构语句中，顺序结构使用最普遍也最简单，本节将介绍顺序结构的基本语法和一些常用函数的使用。

4.1.1 顺序结构流程图和 N-S 流程图

顺序结构采取的是自上而下的思路，整个程序的执行按照书写的顺序从上至下完成，没有分支，程序的流程如图 4-1 所示。

在流程图中，人们发现顺序流程不一定需要，因此又设计了一个新的流程图（N-S 图），它把整个程序放在一个大框中，其中又包含很多小框。从 N-S 图中，可以很清晰地看清程序的整个执行过程。如图 4-2 所示为 if 语句的 N-S 流程图。

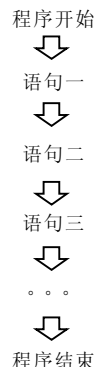


图 4-1 顺序结构程序执行过程

4.1.2 简单的顺序结构程序

本节实例实现三角形面积计算的功能，按照提示输入三角形的底和高，程序自动求出三角形的面积。注意在实际操作中，三角形的底、高、面积不一定是整数，所以接收底和高的变量 a 和 b 必须用 float 型，因此在最后的输出语句中的通配符也应使用 %f 表示。有关通配符的使用会在后面详细说明。

【范例 4.1】三角形面积计算器。

分析：首先考虑必须定义哪些变量，此程序需要用户输入三角形的底和高，因此必须定义两个变量来存放这两个值，而且还需要一个变量来存储三角形的面积。程序根据三角形面积公式计算面积，存入表示面积的变量中，最后输出面积，程序结束。据此算法画出流程图如图 4-3 所示。

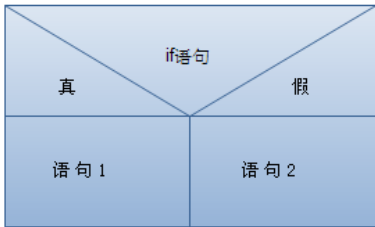


图 4-2 N-S 流程图

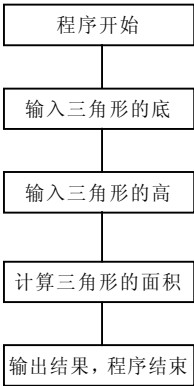


图 4-3 三角形面积计算器程序流程图

范例 4.1 代码实现

```
01  #include<stdio.h>
02  void main()
03  {
04      float a,h,s;
05      printf("请输入三角形的底: \n");          /*提示信息，使界面更友好*/
06      scanf("%f",&a);                          /*接收三角形的底*/
07      printf("请输入三角形的高: \n");
08      scanf("%f",&b);                          /*接收三角形的高*/
09      s=a*h/2;                                  /*计算三角形的面积，结果存入实型变量 s*/
10      printf("三角形的面积为%f",s);            /*输出三角形的面积*/
11  }
```

【代码分析】本程序使用了格式输入输出函数，详细的代码分析如下：

- 第 3~7 行定义变量并初始化，变量的初始化值是来自于 scanf() 函数。
- 第 9 行套用公式计算三角形的面积。

【运行结果】该程序的执行结果如图 4-4 所示。

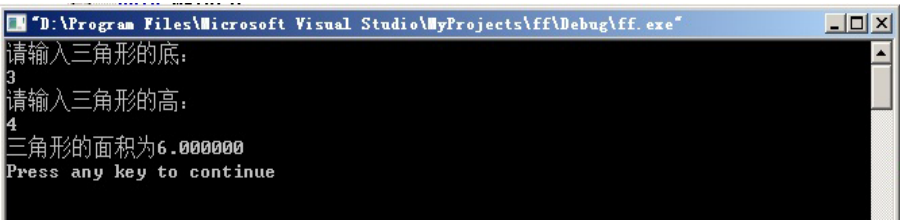


图 4-4 范例 4.1 结果图



4.1.3 了解 C 语言的格式输入、输出函数

输入输出函数是 C 语言编程中最基础的库函数。为了能让计算机完成某个功能，用户必须先输入数据，由计算机处理后输出结果。在 C 语言的初级阶段，输入一般指的是从键盘输入，而输出通常指的是在屏幕上的控制台显示结果。C 语言的输入、输出由 `scanf()` 和 `printf()` 两个内置函数来实现，这两个函数是 C 语言自带的标准库函数，包含在 `stdio.h` 文件中。因此，要使用这两个函数，在程序的第一行必须包含头文件：

```
#include<stdio.h>
```

【范例 4.2】格式输入、输出函数的使用：从键盘输入圆柱体的底面半径和高，求圆柱体的体积。

分析：在程序的开始阶段，需要考虑的问题是，要定义哪些变量，以及这些变量的数据类型。要实现求圆柱体的体积，必须输入圆柱体的底和高，因此需定义两个变量 `r` 和 `h`，还需要定义一个变量 `V`，用来接收计算得到的体积。由于现实中圆柱体的底和高还有面积不都是整数，因此必须将三个变量都定义成 `float` 型。

范例 4.2 代码实现

```
01 #include<stdio.h>
02 void main(){
03     float r,h V;
04     printf("请输入圆柱体底面半径: \n");
05     scanf("%f",&r);
06     printf("请输入圆柱体的高: \n");
07     scanf("%f",&h);
08     V=3.14*r*r*h;
09     printf("圆柱体的体积为%f\n",V);
10 }
```

【代码分析】本例为格式化输入、输出函数简单应用，详细代码分析如下：

- 第 3 行定义了 3 个实型变量。
- 第 4~7 行为接收圆柱体的底和高。
- 第 8 行为计算体积，结果存入变量 `V`。

【运行结果】该程序的执行结果如图 4-5 所示。

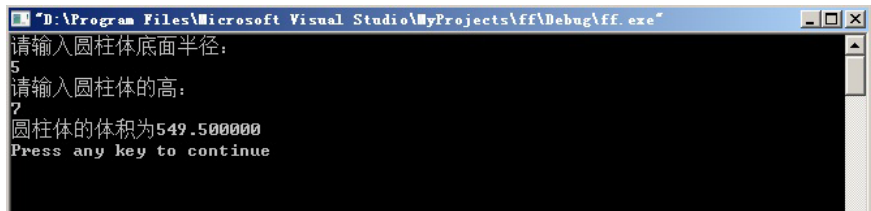


图 4-5 范例 4.2 结果图



注意：printf()函数的变量列表可以是表达式，如实例中的8、9行可以缩写为：

```
printf("圆柱体的体积为%f\n", 3.14*r*r*h);
```

这样变量V无须定义，节省了代码行数，使程序更加简洁，其效果与图4-5一致。



4.2 详解格式输入、输出函数

C语言中包含专门的输入输出函数，其中包括printf()函数、scanf()函数、putchar()和getchar()等函数。本节将通过实例讲解输入、输出函数的使用，注意其通配符和转义字符的使用。

4.2.1 调用scanf()函数实现格式化输入

scanf()函数是格式输入函数，其作用为从键盘读入数据，其类型可为整型、实型、字符型和字符串。

(1) 调用方式：

```
scanf("参数1", 参数2);
```

(2) 参数说明：参数1说明输入数据的格式，参数2说明哪些变量将接受这些数据的值，因此参数1的个数和参数2的个数应是相等的，其位置也应一一对应。下面实例代码的第1、2行定义了3个整型变量，1个实型变量；第3行实现从键盘输入这4个变量的值，以逗号隔开，存入4个变量中。

```
int a,b,c;
float d;
scanf("%d,%d,%d,%f",&a,&b,&c,&d);
```



注意：参数的位置应一一对应，如写三角形的面积计算程序：

```
01#include<stdio.h>
02void main(){
03    float a,h,s;
04    printf("请输入三角形的底: \n");
05    scanf("%f",&a);
06    printf("请输入三角形的高: \n");
07    scanf("%f",&b);
08    s=a*h/2;
09    printf("三角形的底为%f, 高为%f, 面积为%f",s,a,h);
10 }
```

上述代码在编译和运行时不会出错，但是倒数第二行出现了位置不对应的情况，因此程序输出的面积值其实是高的值。这是编程中最为麻烦的事情，编译器会认为程序没有错，程序依然照常运行，但是结果显然错了，在编程中应尽量少出现类似的错误。



(3) 取地址符“&”: 在 scanf()函数第二个参数中出现的“&”为 C 语言的取地址运算符, &a 表示取变量 a 的地址。通过第 3 章变量的定义可知, 声明一个变量, 如 int a;即在内存中为变量 a 开辟一块区域, 用于存放 a 的值, 这块区域的地址就可以用“&a”表示。因此 scanf("%d",&a) 的意思为接收一个整型数, 将其存入变量 a 对应的内存地址中。

(4) 分隔符: 在 scanf()函数的第一个参数中, 逗号的作用是作为分隔符, 用来通知函数上一个数已存入相应的变量, 下一个数开始读入。如果没有逗号, 则以空格作为上一个数结束的标志。如对上段代码的 a、b、c、d 赋值“1, 2, 3, 4”, 若代码为:

```
scanf("%d,%d,%d,%f",&a,&b,&c,&d);
```

控制台赋值时应输入“1, 2, 3, 4”。如果是:

```
scanf("%d%d%d%f",&a,&b,&c,&d);
```

则此时应输入“1 2 3 4”, 中间用空格隔开, 若不隔开, 输入“1234”回车后 a 的值变为 1234, 其他 3 个变量并未赋值。除了用空格外, 也可用回车作为分割符。

(5) 通配符: 说明输入数据的格式, 其前面是一个百分号, 其意思为通知 scanf()函数应输入什么类型的数据, 如%d 代表整型, 那么对应的参数 2 接收值的变量也应为整型, 如下列代码则会报错:

```
float a,b,c;  
float d;  
scanf("%d,%d,%d,%d",&a,&b,&c,&d);
```

因为变量 a、b、c 均为 float 型, 故其对应的通配符应为%f, 而不是%d。正确代码如下:

```
float a,b,c;  
float d;  
scanf("%f,%f,%f,%f",&a,&b,&c,&d);
```

下面通过一个实例来说明格式输入输出函数的调用方法, 注意其中的数据格式及通配符的配套使用。

【范例 4.3】鸡、兔同笼问题: 鸡、兔关在同一个笼子里, 该笼已知有 35 个头, 94 只脚, 要求编程求出鸡兔各有多少只。

分析: 本例是经典数学问题, 可列二元一次方程求解, 将脚用 feet 表示, 头用 head 表示。设鸡 x 只, 兔 y 只则很容易列出方程组:

$$\begin{cases} x+y=\text{head} & \textcircled{1} \\ 2x+4y=\text{feet} & \textcircled{2} \end{cases}$$

①式乘以 2 减去②式得到 $-2y=2h-f$, 进一步运算得到 $y=(f-2h)/2$, 采取同样的方法可得到 $x=(4h-f)/2$ 。以上是本题的算法, 下面的问题是如何用 C 语言来实现该算法。

范例 4.3 代码实现

```
01 #include <stdio.h>  
02 void main(){
```



```
03     int feet=94;
04     int head=35 ;           //定义脚和头的数量
05     int Chicken,Rabbit;     //定义变量存储兔子和鸡的数量，注意变量名易读性
06     Chicken=(4*head-feet)/2; //计算鸡的只数
07     Rabbit=(feet-2*head)/2;  //计算兔子的只数
08     printf("\n 笼中有鸡%d 只，兔%d 只。",Chicken,Rabbit);
09 }
```

【代码说明】本例为鸡兔问题范例，详细代码分析如下所示。

- 第3~4行定义了两个整型变量，一个用于保存脚的数目，另一个用于保存头的数目。
- 第5行，定义了两个整型变量用于保存计算得出的结果。
- 第6、7行，利用分析得出的方程组根据脚和头的数目计算鸡和兔子的只数。

【运行结果】该程序的执行结果如图4-6所示。

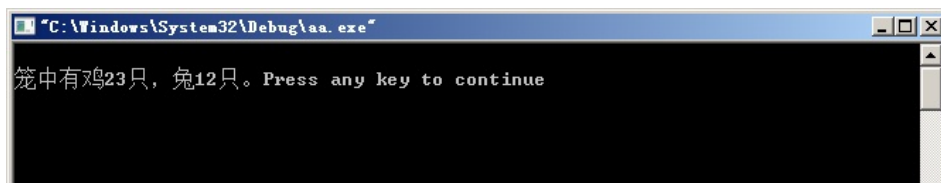


图4-6 范例4.3结果图

4.2.2 调用 printf()函数实现格式化输出

printf()函数是格式输入函数，其作用为从控制台输出数据。与scanf()函数一样，其类型也可为整型、实型、字符型和字符串。

(1) 调用方式：

```
printf("参数1",参数2);
```

(2) 参数说明：

与scanf()函数的参数作用一致，但在参数1可以加入用户想要输出的字符，使程序运行时界面更加友好，下面的代码片段实现初始化4个变量并输出：

```
int a=1;
float b=1.5;
char c='a';
long d=25657;
printf("a是整型数,a的值为: %d\n",a);
printf("b是浮点型数,b的值为: %f\n",b);
printf("c是字符型,c的值为: %c\n",c);
printf("d是长整型,d的值为: %ld\n",d);
```

此段代码运行结果如图4-7所示。

(3) 转义字符：

C语言提供了12个常用转义字符，其作用是为了在输出数据时控制输出的格式，使用方



法如表 4-1 所示。

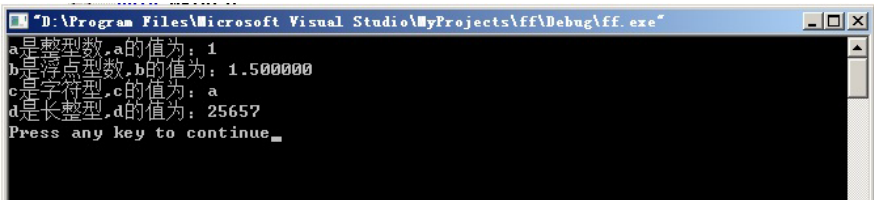


图 4-7 调用 printf()函数实现格式化输出

表 4-1 C 语言转义字符表

转义字符	名 称	描 述
\a	警告	弹出警告
\b	退格	从光标位置退一格
\f	换页	光标移至下一页首行显示
\n	换行	光标移至下一行
\r	回车	光标回车当前行首
\t	水平制表	光标水平移动一定距离
\v	垂直制表	光标垂直移动一定距离
\'	单引号	用于字符
\"	双引号	用于字符串
\?	问号	即 “?”
\\	反斜线	即表示斜杠 “\”
\0	空	用于末尾表示结束

以在程序中输出双引号为例，如写成下列代码：

```
printf("半路杀出个"程咬金"来了");
```

则会出现编译错误，正确格式应如下所示。

```
printf("半路杀出个\"程咬金\"来了");
```

【范例 4.4】大小写字母转换器：将从键盘输入的小写字母变成大写字母，并将其输出至屏幕。

分析：在 C 语言中，字符是以 ASCII 码的形式存放在计算机内存中的，所以可将其当成整型数据处理。C 语言可对整型变量赋字符值，也可对字符型变量赋整型值。同样，在输出时，也可以互换。观察附表 1 可看出，字符 A~Z 的 ASCII 码为 65~90，字符 a~z 的 ASCII 码为 97~122，每个大小写字母的 ASCII 码都相差 32。因此用小写字母的 ASCII 码减去 32 即可得到该字符的大写形式。

范例 4.4 代码实现

```
01 #include <stdio.h>
02 void main(){
03     char a;                                /*定义接收小写字母的变量*/
```



```

04     int b;                                /*定义接收大写字母的变量*/
05     printf("请输入一个小写字母\n");
06     scanf("%c",&a);                        /*变量 a 接收字符值, 注意此时 a 仍为字符型*/
07     b=a-32;                                /*将 a 的 ASCII 码减去 29, 得到大写形式*/
08     printf("字母%c 相应的大写字母是%c",a,b);    /*注意这里用%c 输出了整型变量 b*/
09 }

```

【代码分析】本程序使用了格式输入、输出函数，详细的代码分析如下：

- 第 5、6 行，利用 scanf() 函数接收小写字母。
- 第 7 行，将小写字母减去 32，即实现小写字母与大写字母的转换。

【运行结果】运行结果如图 4-8 所示。

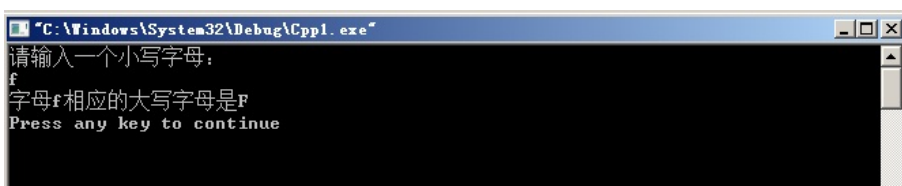


图 4-8 大、小写字符转换器

4.2.3 putchar() 函数

putchar() 函数是字符输出函数，其作用为从控制台输出一个字符，即将一个字符以字符形式显示在屏幕上。

(1) 调用方式：

```
putchar(c);
```

(2) 参数说明：其中 c 为一个字符，可以为 C 语言中的任一字符。如下面的代码，其功能为实现字符的输出，可以比较一下 putchar() 函数和 printf() 函数的不同。

```

char c1='a',c2='b',c3='c';
putchar(c1);
putchar(c2);
putchar(c3);
putchar('\n');
putchar('H');

```

此段代码运行结果如图 4-9 所示。

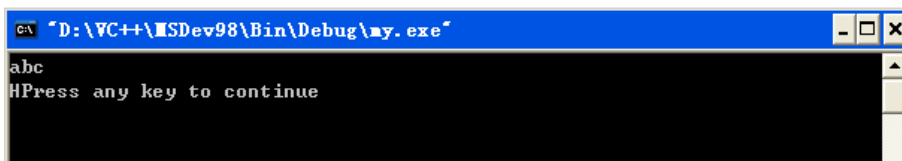


图 4-9 调用 putchar() 函数输出字符

【范例 4.5】编写一个程序，实现 putchar() 函数的简单应用。



分析：每个字符都有着与其对应的 ASCII 码值，`putchar()`函数输出变量时只能输出字符，与 `printf()`函数不同。

范例 4.5 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x;
05      char y;
06      x=99;
07      y='A';
08      putchar(x);
09      putchar('\n');
10      putchar(y);
11      putchar('\n');
12  }
```

【代码分析】本程序使用了格式输入、输出函数，详细的代码分析如下：

- 第 6、7 行，给定义的两个变量赋值。
- 第 8~10 行，利用 `putchar()`函数输出字符至屏幕。

【运行结果】运行结果如图 4-10 所示。

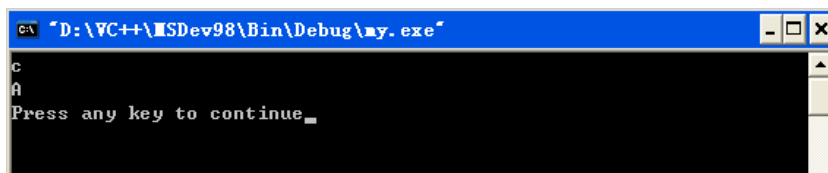


图 4-10 范例 4.5 结果图

4.2.4 `getchar()`函数

`getchar()`函数是字符输入函数，其作用为从控制台接收一个字符，即将一个字符的 ASCII 码值传至控制台中。

(1) 调用方式：

```
getchar(c);
```

(2) 参数说明：其中 `c` 为一个字符，与 `putchar()`函数一样，可以为 C 语言中任一字符。如下面的代码，其功能为实现字符的输入，可以比较一下 `getchar()`函数和 `scanf()`函数的不同。

```
int x;
char c;
x=getchar();
c=getchar();
putchar(x);
printf("\n");
```



```
putchar(c);  
printf("\n");
```

此段代码运行结果如图 4-11 所示。

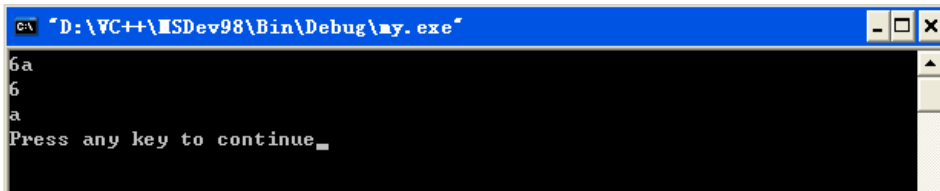


图 4-11 调用 getchar() 函数输入字符

【范例 4.6】编写一个程序，实现 getchar() 函数的简单应用。

分析：getchar() 函数与 scanf() 函数的不同。当 getchar() 函数被调用时，用户输入的字符会被放入键盘缓冲区，当按下回车时，系统就会从缓冲区中每次读入一个字符。

范例 4.6 代码实现

```
01  #include <stdio.h>  
02  void main()  
03  {  
04      int x;  
05      char y;  
06      x=getchar();  
07      y=getchar();  
08      putchar(x);  
09      putchar(y);  
10      putchar('\n');  
11  }
```

【代码分析】本程序使用了格式输入、输出函数，详细的代码分析如下：

- 第 4、5 行定义两个变量 x 和 y。
- 第 6、7 行，调用 getchar() 函数，实现数据的输入。

【运行结果】运行结果如图 4-12 所示。

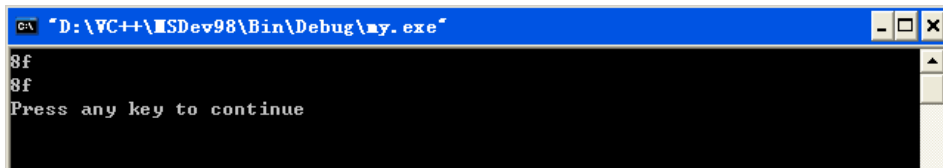


图 4-12 范例 4.6 结果图



注意：第 6 行的 x=getchar()，其功能是实现数字的输入，与 scanf() 函数功能类似，可改写为 scanf(“%d”, &x)。



4.3 本章技术点范例应用

本节为简单的 C 语言应用项目，其详细描述了 C 语言项目开发的一般流程，主要知识点是输入输出函数、输入输出格式控制、通配符的使用。

【范例 4.7】ASCII 码查询系统。

分析：本程序为方便用户查询字符的 ASCII 码，用户输入一个字符，程序显示其 ASCII 码。本程序要用到 `scanf()` 函数和 `printf()` 函数。由字符到 ASCII 码可直接转换，只需将字符变量以整型输出即可，即用通配符 “`%d`” 输出字符，则可得到其 ASCII 码。

范例 4.7 代码实现

```
01 #include <stdio.h>
02 void main(){
03     char c;                                /*定义字符型的变量*/
04     printf("请输入一个字符:\n");
05     scanf("%c",&c);                        /*格式输入字符*/
06     printf("字符%c的ASCII码为%d\n",c,c); /*注意本句代码中数据的转换*/
07 }
```

【代码分析】本例为简单的 ASCII 码查询系统，详细的代码分析如下：

- 第 3~5 行为读入一个字符，存入变量 `c`。
- 第 6 行为输出这个字符的 ASCII 码，注意其中通配符的使用。

【运行效果】运行结果如图 4-13 所示。

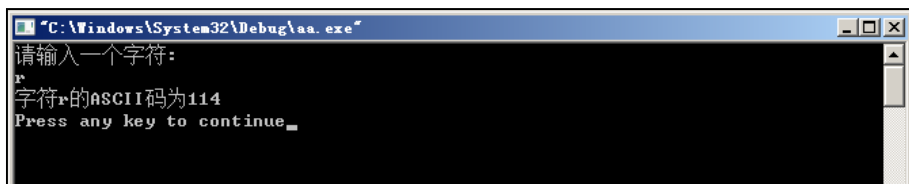


图 4-13 范例 4.7 结果图

C 语言还提供一个 `getch()` 函数，其意思为从键盘读入一个字符，因此对本程序第 5 句写为

```
c=getch();
```

其效果是一样的。



4.4 本章综合练习

【范例 4.8】数值互换：输入两个变量的值，交换输出。

分析：要实现互换变量的值，必须借助第三个变量，先将一个变量的值赋给第三个变量，



然后将第二个变量的值赋给第一个变量，最后将第三个变量值赋给第二个变量，才能实现变量值的互换。

范例 4.8 代码实现

```
01 #include <stdio.h>
02 int main(void)
03 {
04     int a,b,c;                /*注意第三个变量的值也要是相同的数据类型*/
05     printf("请输入变量 a 的值: \n");
06     scanf("%d",&a);
07     printf("请输入变量 b 的值: \n");
08     scanf("%d",&b);
09     printf("交换前两个变量的值分别为 a=%d, b=%d\n",a,b);
10     /*借助 c 值交换 a、b 的值*/
11     c=a;
12     a=b;
13     b=c;
14     printf("交换后两个变量的值分别为 a=%d, b=%d\n",a,b);
15 }
```

【代码分析】本例实现两个数的交换功能，详细分析代码如下所示。

- 第 4 行为定义三个变量，两个变量用于保存输入的数字，另一个变量用于交换数字。
- 第 5~8 行完成数字的输入，即把数据保存至变量 a 和 b 中。
- 第 10~13 行，利用中间变量 c 完成两个数的交换。

【运行结果】程序运行后，第一次输入“5”，第二次输入“89”，可得结果如图 4-14 所示。

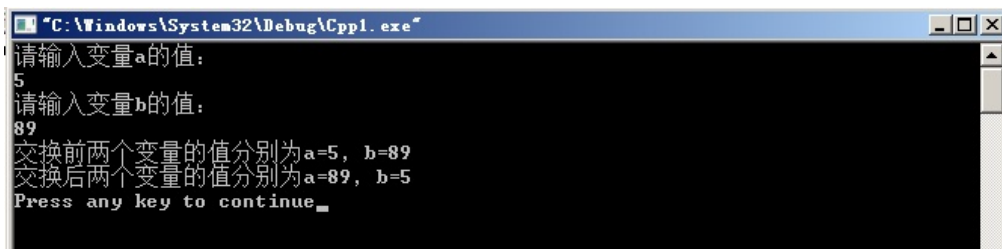


图 4-14 范例 4.8 结果图



4.5 小结

本章讲解了在 C 语言中顺序结构的编程思想，以及如何使用格式输入、输出解决一些实际问题。C 语言还提供了 putchar() 与 getchar() 两个函数来实现输入输出字符，在实际应用中应根据程序的特点来选择。



4.6 习题

一、选择题

1. 设有以下程序，则其输出结果为（ ）。

```
void main()
{
    x=-5321;
    printf("|%03D|",x);
}
```

- A. 语法有误，不能输出 B. |0-532| C. |%03D| D. 5321

【提示】上面的程序中，printf()函数中的输出格式为“|%3D|”，但 C 语言中严格区分大、小写，没有此种格式。它的输出应为|03D|，而与 x 的类型无关。

2. 现有以下程序，当输入 10, 20, 30 时，以下程序的运行结果为（ ）。

```
void main()
{
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    printf("a+b+c=%d",a+b+c);
}
```

- A. a+b+c=50 B. a+b+c=40 C. a+b+c=60 D. 不能确定

【提示】本题利用 scanf()函数给变量 a, b, c 分别赋予 10, 20, 30，利用 printf()函数输出 a+b+c 的结果，即 60。

3. 设有两个数，要将这两个数交换，则下列代码正确的是（ ）。

- A. a=b; b=a; B. b=a; a=b;
C. t=a; a=b; b=t; D. a=b; t=a; b=t;

【提示】要将两个数的值对换，需定义一个中间变量来保存其中的一个数。因为对换两个数的过程要把变量赋给另一个变量，则其值会改变，就不能知道原来该变量的值了。因此交换两个数的值时，必须用一个中间变量保存原来变量的值。如对换 a, b 的值，先用 t=a 保存原来变量 a 的值，再用 a=b 将 b 的值赋给 a，最后将 t 的值赋给 b，即可实现两个数值的对换。

4. 以下程序的运行结果是（ ）。

```
void main()
{
    int x=015;
    printf("%d",++x);
}
```

- A. 13 B. 14 C. 15 D. 20



【提示】本题中 x 被赋值为 015，以 0 开头的数字表示为八进制数字，printf() 函数中输出格式为 %d，因此该程序将八进制数转换为十进制数再输出。

5. 若已知变量 x 和 y 均为 float 类型，则下列输入语句正确的是 ()。

- A. scanf("%f%f", x, y); B. scanf("%f%f", &x, &y);
C. scanf("%d%d", &x, &y); D. scanf("%c%c", &x, &y);

【提示】x 和 y 类型为 float 类型，因此 scanf() 函数通配符应为 %f 格式。

6. 以下程序的运行结果为 ()。

```
void main()
{
    int x=5;
    printf("%\n",x);
}
```

- A. 5 B. % C. 没有输出 D. 编译出错

【提示】本题 printf() 函数中没有固定的输出格式，只有 “%” 符号，因此在编译的过程中会出现错误。

7. 以下程序的运行结果为 ()。

```
void main()
{
    char x='c';
    x--;
    printf("%c,%d",x,x);
}
```

- A. b, 98 B. c, 99 C. b, 99 D. c, 98

【提示】定义一个字符变量 x，初始化其值为字符 c。将变量 x 的值减去 1，即将变量 x 存储字符对应的 ASCII 码值减去 1，因此其数值为 98，表示字符为 b。

8. 以下程序运行时，输入 4 5.6，则其输出结果为 ()。

```
void main()
{
    int a,b;
    scanf("%d%d",&a,&b);
    printf("%d,%d",a,b);
}
```

- A. 4, 5 B. 4, 6 C. 4, 5.6 D. 4, 0

【提示】变量 a 和 b 类型为整型，输出结果也应为整型。

9. 以下程序运行时输入 14, 15, 16，则其输出结果为 ()。

```
void main()
{
    int x,y,z;
    scanf("%d%d%d",&x,&y,&z);
}
```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
printf("x+y-z=%d\n",x+y-z);
}
```

A. 12

B. 13

C. 14

D. 15

【提示】变量 x, y, z 分别赋值为 14, 15, 16, 然后进行简单运算并输出结果。

10. 以下程序运行结果为 ()。

```
void main()
{
    float x=5.5;
    printf("%.1f,%.1f",x,++x);
}
```

A. 5.5, 6.5

B. 6.5, 5.5

C. 6.5, 6.5

D. 5.5, 5.5

【提示】printf()函数中表达式计算是按照从右往左的顺序,因此先计算表达式++x,然后输出变量 x 的值。

二、填空题

1. 以下程序的输出结果为_____。

```
void main()
{
    int a=1,b=3;
    a--;
    printf("%d,%d",a&b++,b);
}
```

【提示】变量 b 的值为 3, a&b++中 a 的值为 0, 因此表达式 b++忽略不计, 不进行运算。

2. 以下程序的输出结果为_____。

```
void main()
{
    int x=4,y=6.322;
    printf("%d",x+y);
}
```

【提示】变量 x 和 y 都为整型, 因此初始化其值分别为 4 和 6, 最终 x+y 的值为 10。

3. 以下程序的运行结果为_____。

```
void main()
{
    int x=1;
    x-=x+=x*x;
    printf("%d",x);
}
```

【提示】变量 x 初始值为 1, 表达式 x-=x+=x*x 中先计算 x=x*x, 然后计算 x=x+x, 最后计算 x=x-x, 因此 x 最终结果为 0。



4. 以下程序的运行结果为_____。

```
void main()
{
    int x,y;
    char c,d;
    float m,n;
    x=35;y=32;
    c='h';d='s';
    m=8.32;n=45.656;
    printf("x=%d,y=%d\n",x,y);
    printf("x=%c,y=%c\n",x,y);
    printf("c=%d,d=%d\n",c,d);
    printf("m=%d,n=%d\n",m,n);
}
```

【提示】本题中定义了两个整型变量 x 和 y ，两个字符变量 c 和 d 及两个浮点型变量 c 和 d 。分别对不同的变量进行赋值，然后输出各个变量相应格式的数，其中注意变量 m 和 n 为浮点型，以 $\%d$ 格式输出其结果为 0。

5. 以下程序运行结果为_____。

```
void main()
{
    int x=5,y=6;
    printf("%d,%d",x,y-x);
}
```

【提示】定义两个整型变量 x 和 y ，输出变量 x 及 $y-x$ 的值。

6. 以下程序运行时输入 $abcd$ ，则其输出结果为_____。

```
void main()
{
    char c;
    scanf("%4c",&c);
    printf("%c",c);
}
```

【提示】定义一个字符变量 c ，输入字符 $abcd$ ，由于字符变量占一位，只能获取一个字符。因此变量 c 中存储的字符为 a 。

7. 以下程序是输入 456789 ，则其运行结果为_____。

```
void main()
{
    int x,y;
    scanf("%3d%2d",&x,&y);
    printf("%d,%d",x,y);
}
```



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】scanf()函数中变量 x 和 y 通配符格式分别为“%3d”和“%2d”，因此变量 x 和 y 从键盘获取的数值分别为 456 和 78。

8. 以下程序输入 120，则其输出结果为_____。

```
void main()
{
    int x;
    scanf("%o",&x);
    printf("%d",x);
}
```

【提示】scanf()函数中通配符格式为“%o”，printf()函数中输出格式为“%d”，因此输出结果为将十六进制数 120 转化为十进制数的结果。

9. 以下程序运行结果为_____。

```
void main()
{
    char x='b'+6;
    printf("%c,%d",x,x);
}
```

【提示】将字符'b'+6 赋值给字符变量 x，因此变量 x 中存储字符为 h，对应 ASCII 码值为 104。

10. 以下程序输入 ab，则其输出结果为_____。

```
void main()
{
    char c1,c2;
    c1=getchar();
    c2=getchar();
    c2=c2+5;
    printf("%c,%c",c1,c2);
}
```

【提示】变量 c1 和 c2 调用 getchar()函数，分别获取字符 a 和 b。将 c2 变量的值加 5 再赋值给变量 c2，最后输出变量 c1 和 c2 的值。

11. 以下程序运行结果为_____。

```
void main()
{
    int n=6,m=8;
    int x=2*n;
    int y=m-1;
    printf("%d,%d",x,y);
}
```

【提示】将变量 n 的值乘以 2 赋值给变量 x，变量 m 的值减 1 赋值给变量 y，然后输出变



量 x 和 y 的值至屏幕。

12. 若要使得以下程序的输出结果为 x=5, y=6, 则在运行时应输入_____。

```
void main()
{
    int a,b;
    scanf("a=%d,b=%d",&a,&b);
    printf("x=%d,y=%d",a++,++b);
}
```

【提示】调用 printf() 函数输出 a++ 的值是先输出变量 a 的值再执行加 1 操作, 而 ++b 是先执行加 1 操作再输出变量 b 的值。

13. 以下程序运行时, 输入 a, d, 则其输出结果为_____。

```
void main()
{
    char c1,c2;
    scanf("%c%c",&c1,&c2);
    printf("%c,%c",c1,c2);
}
```

【提示】上述程序调用 scanf() 函数获取两个字符保存至变量 c1 和 c2 中, 从键盘输入的数据为 a, d, 因此变量 c1 和 c2 中存储的字符为 a 和 d。

14. 以下程序运行结果为_____。

```
void main()
{
    char c='d';
    int a=++c;
    printf("%c",a);
}
```

【提示】字符变量 c 初始值为字符 'd', a=++c 将变量 c 中字符对应的 ASCII 码值加 1 再赋值给变量 a。

15. 以下程序运行结果为_____。

```
void main()
{
    char c1='a',c2='f';
    printf("%c%c\n",c1++,++c2);
}
```

【提示】变量 c1 和 c2 的初始值分别为 'a' 和 'f', 因此 c1++ 和 ++c2 对应的字符为 'a' 和 'g'。

三、编程题

1. 计算一元二次方程的 $x^2+4x+1=0$ 的根, 结果保留两位小数。

【提示】该题目考查的是输入、输出函数及其他库函数的使用, 一元二次方程的求根公式为:



$$X_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

其中要用到开方，C 语言中开方可用库函数 sqrt()实现，sqrt()函数包含在文件<math.h>中，所以程序开始应包含头文件：

```
#include <math.h>
```

因此，求根公式可写为 $x1=(-b+\sqrt{b*b-4*a*c})/2a$ 。由于除法优先级高于加法，因此 $-b+\sqrt{b*b-4*a*c}$ 应加括号，x2 的值同上，结果保留两位小数，则在 printf()函数中通配符应写为%.2f。

【核心代码】

```
#include <math.h>
...
double x1,x2;
    x1=(-1+sqrt(4*4-4*1*1))/2*1;
...
```

2. 已知三角形的三边，计算三角形的面积，结果保留 3 位小数。

【提示】已知三角形三边求面积的公式为：

$$s = \frac{1}{2}(a+b+c)$$

$$\text{area} = \sqrt{s \times (s-a) \times (s-b) \times (s-c)}$$

此题需用户输入三角形的边长，因此需定义三个表示边长的变量。由于有开方运算，所以均定义为 float 型。

【核心代码】

```
float a,b,c,s,area;
scanf("%f,%f,%f",&a,&b,&c);
s=(a+b+c)/2;
area=sqrt(s*(s-a)*(s-b)*(s-c));
```

3. 已知一个圆的周长，编写一个程序求其面积。要求从键盘输入圆的周长，将计算得出的面积输出至屏幕。

【提示】设圆的半径为 r ，则圆的周长 L 和面积 S 公式如下：

$$\begin{cases} 2\pi r = L \\ \pi r^2 = S \end{cases}$$

利用上述的两个公式可得 $S=L^2/4\pi$ ，根据此数学公式编写程序计算即可得出结果，同时周长应注意定义为浮点型，因为周长有可能不为整数。

【核心代码】

```
scanf("%f",&L);
```



```
S=L*L/4*3.14159;
printf("%f",S);
```

4. 编写一个程序，从键盘输入一个字符，实现输出临近该字符的 5 个字符（包括该字符，并以该字符为中心）。

【提示】从键盘输入字符可以利用 `getchar()` 函数来获取该字符，要显示该字符临近的字符可用其 ASCII 值减去一定的数字，利用字符形式输出，即可得到临近的字符。

【核心代码】

```
void main()
{
    char c,c1,c2,c3,c4;
    scanf("%c",&c);
    c1=c-2;
    c2=c-1;
    c3=c+1;
    c4=c+2;
    printf("%c,%c,%c,%c,%c",c1,c2,c,c3,c4);
}
```

5. 编写一个程序，从键盘输入一个梯形的上底、下底和高，求其面积并将结果输出至屏幕。

【提示】设梯形的上底、下底、高分别为 a 、 b 、 h ，则其面积公式如下：

$$s=1/2*(a+b)*h$$

根据上述公式再来编写程序求梯形的面积。

【核心代码】

```
scanf("%f%f%f",a,b,h);
s=1/2*(a+b)*h;
printf("the area is %f",s);
```

6. 从键盘输入一串字符，从该字符串中统计字符 a 的个数。

【提示】利用 `getchar()` 函数逐个地获取字符，直到字符串的末尾。同时设置一个初始值为 0 的计数器，将获取的字符与 a 进行比较，若相等则计数器加 1，否则不变。最后输出计数器的数值，即为 a 的个数。

【核心代码】

```
int count=0;
char c;
while((c=getchar())!='\n')
{
    if(c=='a')
        count++;
}
```

7. 从键盘输入 3 个数，编写一个程序计算其总和及平均值。



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】将从键盘输入的 3 个数保存至 3 个变量中，然后设置一个变量 s ，初始值为 0，用来计算 3 个数的累积和，最后除以 3 求得其平均值。

【核心代码】

```
sum=0;
scanf("%d%d%d",&a,&b,&c);
sum=a+b+c;
aver=sum/3;
```

8. 编写程序将时间转化为秒后输出，例如 8 时 20 分 45 秒转化过程为 $8*3600+20*60+45$ 。

【提示】将时间转化为秒，其转换过程为小时乘以 3600 加上分钟乘以 60 再加上秒数，最后即为所求结果。

【核心代码】

```
scanf("%d%d%d",&hour,&minute,&second);
s=hour*3600+minute*60+second*45;
printf("%d\n",s);
```

9. 从键盘输入两个数，交换两个数的值，再输出至屏幕。

【提示】交换两个数的大小，需要一个临时变量。先用临时变量保存一个数的大小，然后将另一个变量的值赋给该变量，在将临时变量中的值赋给另外一个变量，即可实现值的交换功能。

【核心代码】

```
scanf("%d%d",&x,&y);
t=x;
x=y;
y=t;
```

10. 现有函数 $y=5x+3$ ，从键盘输入一个数，输出对应的 y 值。

【提示】从键盘获取数据通过 `scanf()` 函数实现，然后将 $5x+3$ 的值赋给变量 y ，最后调用 `printf()` 函数输出结果至屏幕。

【核心代码】

```
scanf("%d",&x);
y=5*x+3;
printf("%d",y);
```

11. 从键盘输入一个小写字母，要求将其转换为大写字母后输出。

【提示】小写字母取值范围为 $a\sim z$ ，将其转换为大写字母，只需将其 ASCII 码值减去 32 即可。

【核心代码】

```
scanf("%c",&c)
if(c>='a'&&c<='z')
c=c-32;
```

第 5 章 条件结构程序设计

在 C 语言的算法中，任何一个表达式都要首先判断其值真假，即判断是否满足该条件，从而决定是否执行该语句。条件结构程序设计主要是通过对条件进行判断来执行的，其与日常生活中的一些问题息息相关，因此学会并掌握条件结构程序设计至关重要。在本章中，将会讲解条件结构化程序的设计及其使用。

本章主要涉及的知识点有：

- 结构化程序设计思想；
- 条件结构程序设计；
- C 语言中的逻辑值（真、假）；
- 结构化程序设计的应用；
- 运算符。



5.1 条件结构简介

本节主要介绍条件结构的基本概念及一些简单的应用和 if 语句的几种形式，这些概念都是 C 语言的基础，因此读者应认真掌握本节的知识。

5.1.1 if 单分支形式

if 单分支用来判断条件，若条件满足，执行 if 后的语句，否则跳过 if 后的语句。其形式如下：

```
if(表达式) 语句
```

例如：

```
if(x>0) printf("%d\n",x);
```

上例中判断 x 是否大于 0，如果大于 0 则输出 x，反之跳过该语句。



注意：表达式条件要用()包围，以及语句的末尾要加“；”表示语句的结束，否则编译过程中会显示相应的错误信息。

流程图和 N-S 图如图 5-1 所示。

if 后的语句称为 if 子句，其执行步骤如下所示。

- (1) 判断表达式的真值。
- (2) 若表达式的逻辑值为真，则执行 if 语句。
- (3) 若表达式的逻辑值为假，则跳过 if 语句执行其后面的语句。



图 5-1 流程图和 N-S 图

【范例 5.1】输入两个整数，输出两个整数中的大者。

分析：首先定义两个整型变量，用于保存从键盘输入的数，在 if 语句中比较两个数的大小并输出其中的大者。

范例 5.1 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int a,b;                /*定义两个整型变量*/
05      printf("请输入两个整数: "); /*提示输入两个整数*/
06      scanf("%d%d",&a,&b);      /*用 scanf()函数进行数字的输入，中间用空格分开*/
07      if(a>b) printf("%d\n",a); /*若 a 大于 b 输出 a*/
08      if(a<b) printf("%d\n",b); /*若 b 大于 a 则输出 b*/
09  }
```

【代码分析】本例为 if 语句的一个简单应用，详细代码分析如下：

- 第 6 行，调用 scanf()函数进行数字的输入。
- 第 7、8 行，通过 if 语句判断变量 a 和 b 的大小，其中若变量 a 大于 b 输出 a，若变量 a 小于 b 则输出变量 b。

【运行结果】该程序的执行结果如图 5-2 所示。

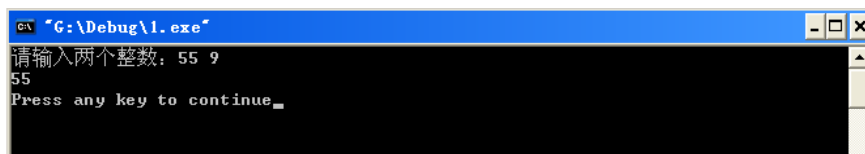


图 5-2 范例 5.1 结果图



注意：调用 scanf()函数进行数字的输入时，需用空格把两个数给分开，否则会造成数据读取的错误。

【范例 5.2】输入一个实数 x，求其绝对值 y。

分析：首先定义两个浮点型变量，一个用于保存输入的实数，一个用于保存其绝对值。令 $y=x$ ，判断 x 是否大于 0，若变量 x 小于 0，令 $y=-x$ 即可。

范例 5.2 代码实现

```
01  #include <stdio.h>
02  void main()
```



```
03 {
04     float x,y;                /*定义两个浮点型变量*/
05     printf("请输入 x 的值:"); /*提示输入 x 的值*/
06     scanf("%f",&x);          /*用 scanf() 函数进行数字的输入*/
07     y=x;                      /*把 x 的值赋给 y*/
08     if(x<0) y=-x;             /*如果 x 为负数, 把 x 的相反数赋给 y*/
09     printf("x=%f y=%f\n",x,y); /*输出 x, y 的值*/
10 }
```

【代码分析】本例也是 if 的一个简单应用，详细代码分析如下：

- 第 7、8 行，首先令 $y=x$ ，若 x 为负数，则令 $y=-x$ ，即求得 x 的绝对值。

【运行结果】该程序的执行结果如图 5-3 所示。

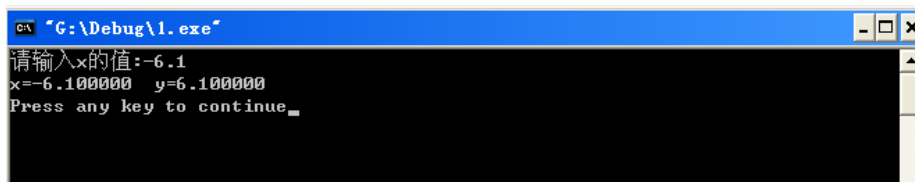


图 5-3 范例 5.2 结果图



注意：第 7、8 行可改为 $y=\text{fabs}(x)$ 来求 x 的绝对值，即调用 $\text{fabs}()$ 函数直接求 x 的绝对值，该函数包含在 math.h 头文件中，这样可以使程序代码简洁一些，但在使用之前必须先包含该头文件。

5.1.2 if-else 双分支形式

if-else 形式与 if 形式相似，也是对条件进行判断并执行相应的语句。if-else 结构中若 if 后表达式条件不满足，则执行 else 后的语句。其形式如下：

```
if(表达式)    语句 1
else          语句 2
```

此结构在 C 语言中应用较频繁，也较易理解和掌握。其中语句 1、语句 2 可为简单语句也可为复合语句，如是复合语句必须使用“{}”包围，否则只能执行第一句。若语句 1、语句 2 是复合语句，那么将是后面会讲到的嵌套 if。

if 后的语句称为 if 子句，else 后的语句称为 else 子句。



注意：在 if-else 结构的使用过程中，else 要与 if 配套使用不能单独使用。

if-else 执行的一般步骤如下：

- (1) 判断表达式的真值。
- (2) 若表达式的逻辑值为真，则执行 if 子句（语句 1）。
- (3) 若表达式的逻辑值为假，则执行 else 子句（语句 2）。

流程图和 N-S 图，如图 5-4 所示。

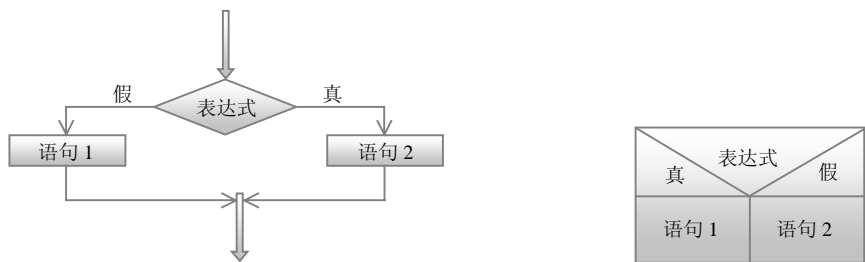


图 5-4 if-else 结构流程图和 N-S 图

【范例 5.3】输入一个数，判断这个数是否能被 5 整除，如果能被整除则输出 “yes”，否则输出 “no”。

分析：判断一个数是否能被 5 整除，可以判断其除以 5 的余数是否为 0。若为 0 则表示该数可以被 5 整除，否则不能被 5 整除。

范例 5.3 代码实现

```
01  #include <stdio.h>                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int a;                        /*定义一个整型变量*/
05      printf("请输入 a 的值:");      /*提示输入 a 的值*/
06      scanf("%d",&a);                /*用 scanf 函数来进行数字的输入*/
07      if(a%5==0) printf("yes\n");    /*如果 a 除 5 余数为 0，输出 yes*/
08      else      printf("no\n");      /*如果 a 除 5 余数不为 0，输出 no*/
09  }
```

【代码分析】本程序是 if-else 的一个应用范例，详细代码分析如下：

- 第 7、8 行，判断整型变量 a 除以 5 的余数，看其是否为 0 来判断是否能够被 5 整除，并进行相应的输出。

【运行结果】该程序的执行结果如图 5-5 所示。

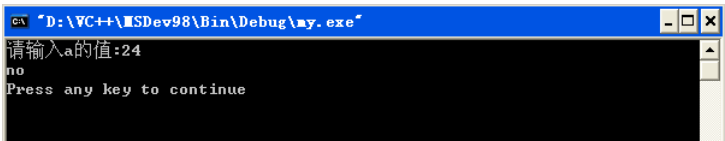


图 5-5 范例 5.3 结果图



5.2 多重 if

if 语句主要有 3 种形式，前面已经讲解过 if 单分支形式和 if-else 双分支形式，接下来讲解多重 if 形式，也称为 else-if 形式。其一般形式如下：

```
if(表达式 1)        语句 1
else if(表达式 2)    语句 2
```

```
else if(表达式 3)      语句 3
else if(表达式 4)      语句 4
...
else if(表达式 n-1)    语句 n-1
else                   语句 n
```

其中语句 1 至语句 n 可为简单语句也可为复合语句，复合语句必须使用“{}”包围，表达式可为符合 C 语言语法的任意表达式。其中 if 为整个结构的开始，else 为整个结构的结束。

else-if 执行的一般步骤如下：

- (1) 判断表达式 1 的值，若其逻辑值为真，则执行语句 1，否则执行步骤 2。
- (2) 判断表达式 2 的值，若其逻辑值为真，则执行语句 2，否则执行步骤 3。
-

如果 else 上面的表达式都为假，则执行语句 n ，并结束该结构。
其流程图如图 5-6 所示。

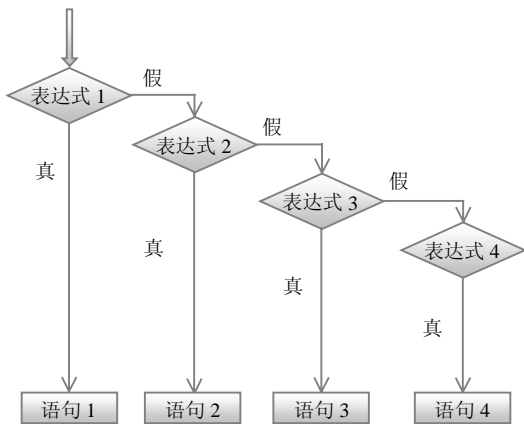


图 5-6 else-if 流程图

从该流程图可以很清楚地看到多重 if 执行的整个过程。

【范例 5.4】输入学生成绩，范围为 0~100 分。若成绩在 90 分以上，则打印出“A”级，并输出“优秀”；若成绩在 80~90 分之间输出“B”级并打出“良好”；若成绩在 70~80 分之间输出“C”级并打出“中等”；若成绩在 60~70 分之间输出“D”级并打出“及格”；若成绩低于 60 分输出“E”级并打出“不及格”。

分析：该程序可用 else-if 结构来确定分数的范围，并根据其确定的范围进行相应的输出。

范例 5.4 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int score;                      /*定义一个整型变量用于保存分数*/
05      printf("输入分数:");
06      scanf("%d",&score);
07      if(score>=90)                  /*判断 score 是否大于 90*/
```



```
08  printf("A 优秀\n");           /*满足条件则输出 A 优秀*/
09  else if(score>=80)           /*隐含了 score 小于 90 分的条件*/
10  printf("B 良好\n");         /*满足条件则输出 B 良好*/
11  else if(score>=70)           /*隐含了 score 小于 80 分的条件*/
12  printf("C 中等\n");         /*满足条件则输出 C 中等*/
13  else if(score>=60)           /*隐含了 score 小于 70 分的条件*/
14  printf("D 及格\n");         /*满足条件则输出 D 及格*/
15  else                         /*隐含了 score 小于 60 分的条件*/
16  printf("E 不及格\n");       /*满足条件则输出 E 不及格*/
17  }
```

【代码分析】本程序是用 else-if 来确定分数的等级，详细代码分析如下：

- 第 4 行定义了一个 score 整型变量用于保存分数。
- 第 7~16 行用 else-if 结构来确定分数的范围并用 printf() 函数进行相应的输出。

【运行结果】该程序的执行结果如图 5-7 所示。

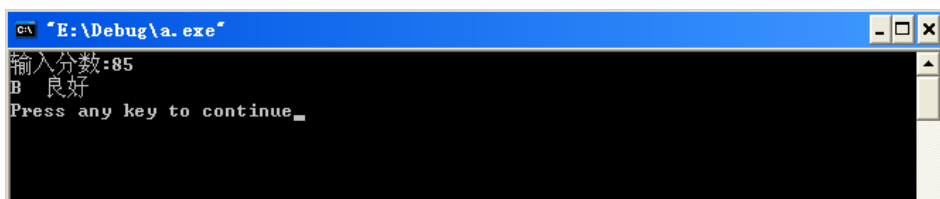


图 5-7 范例 5.4 结果图



提示：scanf() 函数中的“&”符号要加进去，不然结果会不正确，造成意想不到的错误。



5.3 嵌套 if

嵌套 if 是指 if 语句中包含一个或多个 if 语句。其格式如下：

```
if(表达式 a)
    if(表达式 b)    语句 1
    else            语句 2
else
    if(表达式 c)    语句 3
    else            语句 4
```

嵌套 if 书写要规范，要有缩进，这样看上去会很清晰。else 总是与它最近的 if 语句配套。

【范例 5.5】输入三个数 x, y, z，实现从大到小地排列，用嵌套 if 实现。

分析：首先要清楚三个数总共会有 6 种关系，可先在外层比较两个数的大小，再在内层比较第三个数的大小。



范例 5.5 代码实现

```
01 #include <stdio.h>
02 void main()
03 {
04     int x,y,z;                /*定义三个整型变量用于保存数字*/
05     printf("输入 x y z 的值:"); /*提示输入数字*/
06     scanf("%d%d%d",&x,&y,&z);    /*用 scanf() 函数进行分数的输入, 中间用空格隔开*/
07     printf("x y z 从大到小排列为:"); /*提示文字*/
08     if(x>=y)                  /*嵌套 if*/
09     { if(z>=x)                printf("%d %d %d\n",z,x,y);          /*内嵌 if 语句*/
10       else if(z<y)            printf("%d %d %d\n",x,y,z);
11       else                    printf("%d %d %d\n",x,z,y);
12     }
13     else
14     { if(z>=y)                printf("%d %d %d\n",z,y,x);          /*内嵌 if 语句*/
15       else if(z<x)            printf("%d %d %d\n",y,x,z);
16       else                    printf("%d %d %d\n",y,z,x);
17     }
18 }
```

【代码分析】本程序是用嵌套 if 来确定三个数的大小，详细代码分析如下：

- 第 4 行定义了三个整型变量用于保存数字。
- 第 8~17 行为内嵌 if 结构，外层通过对变量 x, y 比较大小，内层通过把变量 z 与 x, y 比较大小从而确定三个数的大小。

【运行结果】该程序的执行结果如图 5-8 所示。

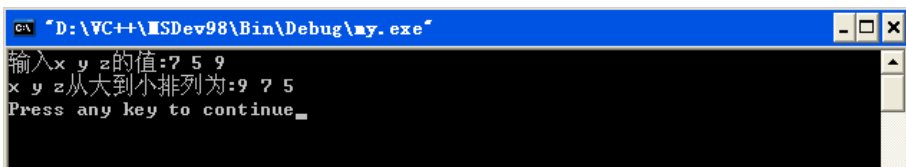


图 5-8 范例 5.5 结果图



5.4 switch 结构

人们有时会处理一些多分支的事务，如学生成绩管理、资料管理等。虽然这些可用 if 语句或嵌套 if 语句来实现，但往往显得较烦琐。因此在 C 语言中，引进了另外一种结构即 switch 结构，其作为一种多分支的选择结构，能够很好地处理多分支的线路问题。switch 结构的一般形式如下：

```
switch(表达式)
{
    case 常量表达式 1: 语句 1
    case 常量表达式 2: 语句 2
```




```
case 常量表达式 3: 语句 3
...
case 常量表达式 n: 语句 n
default:          语句 n+1
}
```

其中,常量表达式必须为常量,语句可为简单语句,也可为复合语句。**Switch**、**case** 和 **default** 都为关键字。一般在 **switch** 语句后要加 **break** 语句,以退出 **switch** 结构,否则程序会一直执行到 **default** 才退出 **switch** 结构。其中 **case** 事件的顺序对程序并没有影响。

switch 结构执行的一般步骤如下:

(1) 计算 **switch** 中表达式的值。

(2) 比较常量表达式和 **switch** 表达式的值,若常量表达式的值和 **switch** 表达式的值相等则执行其后面的语句,否则执行 **default** 后的语句。

其流程图如图 5-9 所示。

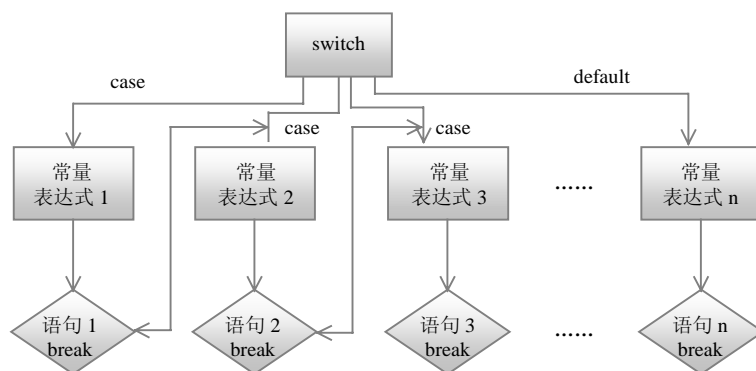


图 5-9 switch 结构流程图

【范例 5.6】设计一个简单的计算器,可以进行简单的+、-、*、/运算。

分析:定义两个变量用于保存要操作的数,然后定义一个变量用于保存操作符,利用 **switch** 结构根据相应的操作符进行相应的运算并输出。

范例 5.6 代码实现

```
01 #include <stdio.h>
02 void main()
03 {
04     int a,b,d;                                /*定义三个整型变量*/
05     char c;                                  /*定义一个字符用于保存操作符*/
06     printf("输入 a,b 的值以及要进行的操作中间不用空格分开:"); /*提示输入字符*/
07     scanf("%d%c%d",&a,&c,&b);                /*用 scanf() 函数进行分数的输入*/
08     switch(c)                                /*switch 结构*/
09     { case '+': d=a+b;break;                  /*若操作符为+, 则相加*/
10       case '-': d=a-b;break;                  /*若操作符为-, 则相减*/
11       case '*': d=a*b;break;                  /*若操作符为*, 则相乘*/
12       case '/': d=a/b;break;                  /*若操作符为/, 则相除*/
```



```

13     default: printf("输入的操作有误\n");           /*若以上条件都不满足则输出该语句*/
14     }
15     printf("%d %c %d=%d\n",a,c,b,d);
16     }

```

【代码分析】本程序是用 switch 结构构造了一个简单的计算器，详细代码分析如下：

- 第 4 行定义了三个整型变量，两个变量用于保存输入的数字，一个变量用于保存计算结果。
- 第 5 行定义了一个字符型变量用于保存键入的操作符。
- 第 8~14 行为 switch 结构，对输入的操作符及常量表达式进行比较，并进行相应的操作。

【运行结果】该程序的执行结果如图 5-10 所示。

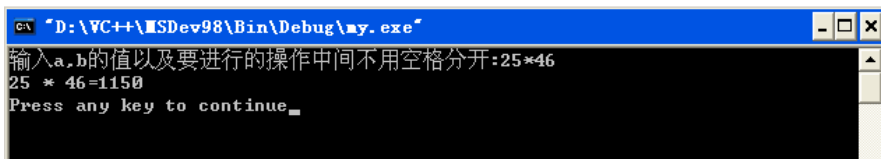


图 5-10 范例 5.6 结果图



注意：本题中 scanf()函数不必要用空格分开，因为“%d”和“%c”的格式不同，编译器会自动根据其类型不同，读取相应的数据保存至变量中。



5.5 实战项目

【范例 5.7】现有一企业发放的奖金根据年工资提成，其提成如下所示。

- 工资低于或等于 3 万元时，奖金可提 8%。
- 工资高于 3 万，低于或等于 7 万奖金可提 5%。
- 工资高于 7 万，低于或等于 10 万可提 3%。
- 工资高于 10 万可提 1%。

从键盘输入年工资，求奖金可提成多少。

分析：可用数轴来分界，定位。定义时需把工资定义成长整型，以防止超过整型的最大上限，数据发生溢出。

范例 5.7 代码实现

```

01     #include <stdio.h>
02     void main()
03     {
04         long int a;           /*定义一个长整型变量用于保存年工资*/
05         int b;               /*定义一个整型变量用于保存奖金*/
06         printf("请输入年工资:"); /*提示输入年工资*/
07         scanf("%ld",&a);      /*用 scanf() 函数进行工资的输入*/
08         if(a<=30000)         /*判断年工资是否小于 3 万*/

```



```
09  b=a*0.08;           /*如工资小于 3 万提取 8%*/
10  else if(a<=70000)    /*隐含了 a 大于 3 万的条件*/
11  b=a*0.05;           /*如工资大于 3 万小于 7 万提成 5%*/
12  else if(a<=100000)   /*隐含了 a 大于 7 万的条件*/
13  b=a*0.03;           /*如工资大于 7 万小于 10 万提成 3%*/
14  else                 /*隐含了 a 大于 10 万的条件*/
15  b=a*0.01;           /*如工资大于 10 万提成 1%*/
16  printf("奖金为%d\n",b); /*输出奖金的数目*/
17  }
```

【代码分析】本程序是 else-if 结构的简单应用，详细代码分析如下：

- 第 4 行，把年工资定义为长整数，因为年工资会超过整型的最大上限。
- 第 8~15 行是 else-if 结构，首先根据开头的 if 语句判断真假，若为真则执行 if 子句，退出 else-if 结构；若为假则接着执行下面的 else-if 子句，再判断真假继续执行，直到有表达式为真为止；若 else 上的表达式都为假，即年工资大于 10 万，则执行 else 子句。

【运行结果】该程序的执行结果如图 5-11 所示。

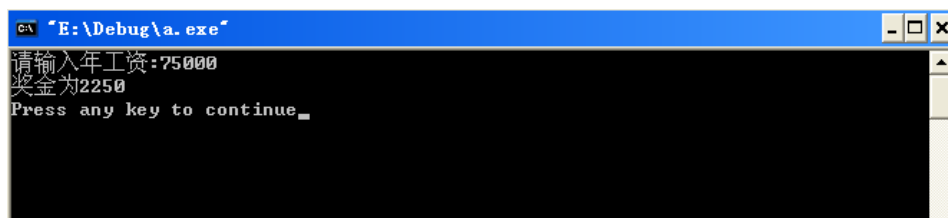


图 5-11 范例 5.7 结果图

【范例 5.8】输入学生成绩的等级（A、B、C、D、E），输出相应的评价文字（优秀、良好、中等、及格、不及格）。

分析：用 switch 结构中的常量表达式与 switch 表达式相比较，进行相应的输出。其流程图如图 5-12 所示。

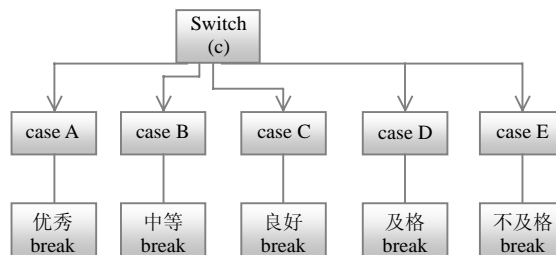


图 5-12 范例 5.8 流程图

范例 5.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04  char c;                               /*定义一个字符型变量*/
```



```

05  printf("输入一个字符:");          /*提示输入字符*/
06  c=getchar();                      /*用 getchar()函数输入字符*/
07  switch(c)                         /*switch 结构*/
08  { case 'A' : printf("优秀\n");break; /*若字符为 A, 输出优秀*/
09    case 'B' : printf("良好\n");break; /*若字符为 B, 输出良好*/
10    case 'C' : printf("中等\n");break; /*若字符为 C, 输出中等*/
11    case 'D' : printf("及格\n");break; /*若字符为 D, 输出及格*/
12    case 'E' : printf("不及格\n");break; /*若字符为 E, 输出不及格*/
13    default: printf("输入有误\n");    /*若以上条件都不满足则输出该语句*/
14  }
15  }

```

【代码分析】本程序是 switch 结构的简单应用，详细代码分析如下：

- 第4行定义了一个字符型变量 c 用于保存输入的字符。
- 第6行，调用 getchar() 函数从键盘获取字符保存至字符变量 c 中。
- 第7~14行是 switch 结构，通过把输入的字符与常量表达式进行比较来进行相应的输出。若变量 c 的值为 A，则输出“优秀”；若其值为 B，则输出“良好”；若其值为 C，则输出“中等”；若其值为 D，则输出“及格”；若其值为 E，则输出“不及格”，否则输出“输入有误”至屏幕。

【运行结果】该程序的执行结果如图 5-13 所示。

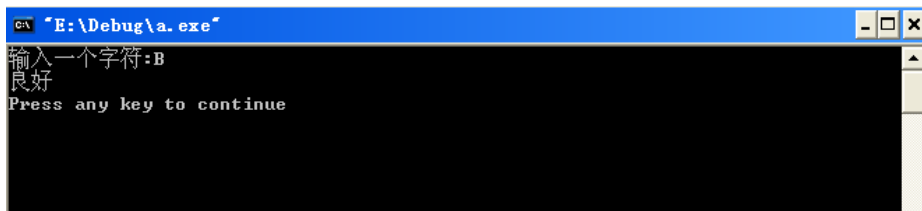


图 5-13 范例 5.8 结果图



技巧：getchar()函数的功能是用来获取一个字符，该函数也可以用 scanf()函数替代。switch 结构中加 break 语句来跳出该结构，否则会输出所有的结果，最后一个 default 可以不加 break 语句。

【范例 5.9】输入三角形三边的大小，判断这三条边是否能构成三角形。若能构成三角形输出该三角形的面积，否则输出“不能构成三角形”的信息。

分析：定义三个变量保存三边的大小，并判断三边的关系，看其是否能构成三角形，若能构成三角形则计算其面积。计算面积按照如下公式：

```

a=(x+y+z)*0.5
s=sqrt(a*(a-x)*(a-y)*(a-z));

```

范例 5.9 代码实现

```

01  #include <stdio.h>
02  #include <math.h>

```



```
03 void main()  
04 {  
05     float x,y,z,a,s;                /*定义 5 个浮点型变量*/  
06     printf("请输入三边:");          /*提示输入边大小*/  
07     scanf("%f%f%f",&x,&y,&z);        /*用 scanf() 函数进行三边的输入*/  
08     if((x+y)>z&&(x+z)>y&&(y+z)>x)    /*判断三边是否满足构成三角形的条件*/  
09     { a=(x+y+z)*0.5;                /*套用公式*/  
10       s=sqrt(a*(a-x)*(a-y)*(a-z));  /*套用公式*/  
11       printf("该三角形的面积为%f\n",s); /*输出三角形面积*/  
12     }  
13     else                             /*若三边不能构成三角形*/  
14     printf("不满足三角形条件\n");    /*输出提示语句*/  
15 }
```

【代码分析】本程序通过三边构造三角形并计算面积，详细代码分析如下：

- 第 2 行包含 `math.h` 头文件，把一些基本的熟悉函数包含了进来，例如下面用到的 `sqrt()` 函数就是包含在 `math.h` 头文件中的。
- 第 5 行定义了 5 个浮点型变量，其中 3 个用于保存三边大小，2 个用于计算和保存三角形的面积。
- 第 7 行用 `scanf()` 函数进行三边的输入，`scanf()` 函数中的 “&” 符号一定要加上，三边的类型为浮点型，格式应为 `f%`，中间用空格分开。
- 第 8~14 行判断三边是否满足构成三角形的条件，即两边之和是否大于第三边。若满足条件则套用公式计算其面积并输出，若不满足条件则提示不能构成三角形。

【运行结果】该程序的执行结果如图 5-14 所示。

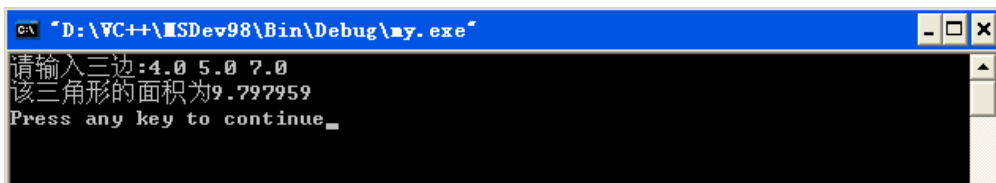


图 5-14 范例 5.9 结果图



技巧：`sqrt()` 函数包含在 `math.h` 头文件中，要引用该函数，必须先包含此 `math.h` 头文件。



5.6 小结

在本章开始讲解了 `if` 的三种形式，即 `if` 单分支、`if` 双分支和多重 `if`，后面则讲述了 `switch` 结构的概念及应用，以及算术符的概念及其表达式，这些在 C 语言中被频繁使用，因此掌握这些知识对学好 C 语言将会有很大的帮助。



5.7 习题

一、选择题

1. 对于下面这个程序, () 选项说法正确。

```
#include <stdio.h>
void main()
{ int a,b;
  scanf("%d%d",&a,&b);
  if(a<b) printf("%d",a);
  else printf("%d",b);
}
```

- A. 该程序是输出两个数中的大者
- B. 该程序有错, 不能运行
- C. 输入两个数时, 不用在中间加空格
- D. 该程序输出两个数的小者

【提示】该题考察 scanf() 函数的基本使用和 if 语句的判断, 通过题目不难判断出答案。

2. 下列关于程序的运行结果正确的是 ()。

```
#include <stdio.h>
void main()
{ int x,y,z;
  x=1;
  y=5;
  z=10;
  if(x<y) x=z;y=x;
  z=y;
  printf("%d,%d,%d",x,y,z);
}
```

- A. x=1, y=10, z=5
- B. x=10, y=10, z=5
- C. x=10, y=10, z=10
- D. x=5, y=10, z=5

【提示】该题考察变量赋值的概念, 即把=号右边的式子赋给左边, 若抓住这个概念, 此题就不会很复杂了。观察哪个式子赋给哪个变量, 同时注意变量赋值后的变化即可。

3. 在下面的选项中, 与 if (A) 中的 (A) 等价的表达式是 ()。

- A. (A==0)
- B. (! A=0)
- C. (A!=0)
- D. 都不正确

【提示】通过 if(A) 中的 A 条件进行判断, 若该条件满足, 即 A 为真, 便可判断出正确的选项。

4. 下面程序输出结果正确的是 ()。

```
#include <stdio.h>
void main()
```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
{ int a=0,b;
  if(a<b) printf("a<b");
  else printf("a>=b");
}
```

A. 程序有错，不能运行

B. 输出 a<b

C. 输出 a>b

D. 输出结果不能确定

【提示】该题中变量 b 并没有赋初始值，所以 b 的值不能确定，不能和 a 进行判断比较。

5. 有以下程序，输出结果为（ ）。

```
#include <stdio.h>
void main()
{ int x=10,a=2,z=1;
  switch(a)
  { case 0:z++;break;
    case 1: z--;break;
    case 2:z=x;break;
    default:z=0;
  }
  printf("%d",z);
}
```

A. z=0

B. z=1

C. z=10

D. z=2

【提示】该题考察 switch 结构的应用，对条件进行相应的判断即可。

6. 下列程序中正确的是（ ）。

A.

```
#include <stdio.h>
void main()
{ int a=1,b=10;
  switch(a)
  { case b:a++;break;
    case b+1:a--;break;
    case b-1:break;
  }
}
```

B.

```
#include <stdio.h>
void main()
{ int a=1,b;
  switch(a)
  { case a==1:b=a;break;
    case a>1:a--;break;
    case a<1:a++;break;
  }
}
```

C.

```
#include <stdio.h>
void main()
{ int a=1,b=10;
  switch(a)
  { case 1:a++;break;
    case 3-2:a--;break;
    case 10:break;
  }
}
```

D.

```
#include <stdio.h>
void main()
{ int a=1,b;
  switch(a)
  { case 1:a++;break;
    case 2:a--;break;
    case 10:break;
  }
}
```

【提示】该题目也是考察 switch 结构的应用，主要看其内容是否为 switch 结构的语法。switch 结构中的条件不能为变量，必须是常量或常量表达式，条件的值不能重复。



7. C语言中能正确表达关系“ $x \geq 5$ ”或“ $x < 1$ ”的表达式为()。

A. $x \geq 5 | x < 1$

B. $x \geq 5 \text{ or } x < 1$

C. $x \geq 5 \&\& x < 1$

D. $x \geq 5 || x < 1$

【提示】该题考察了一些基本的关系运算符的概念。

8. 下列程序的运行结果正确的是()。

```
#include <stdio.h>
void main()
{ int x=3,y=1,z=1;
  if(x=y-z)
printf("****");
  else
  printf("---");
}
```

A. 输出结果为“****”

B. 输出结果为“---”

C. 该程序有错，不能运行

D. 都不对

【提示】该题考察 if-else 和逻辑值的应用。如 $y-z$ 的值为 0，即为逻辑假，if 中的条件不满足跳过 if 语句，执行 else 后的语句。

9. 下列关于程序的输出结果正确的是()。

```
#include <stdio.h>
void main()
{ int x=1,y=0,z=0;
  switch(x)
  { case 0:z++;
  case 1:y++;
  case 2:y++;z++;
  }
  printf("y=%d,z=%d",y,z);
}
```

A. $y=2, z=2$

B. $y=2, z=1$

C. $y=1, z=1$

D. $y=1, z=2$

【提示】该题考察 switch 结构的应用。以 x 作为 switch 结构的入口， x 初始值为 1，从 case 1 开始向下执行，因 case 语句后没有 break 语句，不会退出该结构，会继续执行直到遇到该结构的末尾或遇到 break 语句。

10. 下列程序运行后，t 的值为()。

```
#include <stdio.h>
void main()
{ int x=0,y=0,z=0;
  int t=30;
  if(!x) t++;
  else if(y) ;
  if(z) t=0;
```




```
else t=40;
}
```

- A. 0
B. 40
C. 31
D. 30

【提示】本题主要考察 if-else 及 else-if。在本题中只要抓住 else 总是与其最近的 if 配对即可得出正确答案。

11. 若 $y=(x>1?3:x<1?2:1)$ ，下列选项中与该语句意义一样的选项是（ ）。

- A.

```
if(x>1)    y=3;
else if(x<1) y=2;
else      y=1;
```
- B.

```
if(x>1)    y=3;
if(x<1)    y=2;
else      y=1;
```
- C.

```
y=3;
if(x<1)    y=2;
else      y=1;
```
- D.

```
y=1;
if(x>1)    y=3;
if(x<1)    y=2;
```

【提示】本题考察三目运算符和 if 结构的概念。 $y=(x>1?3:x<1?2:1)$ 按照从右至左的原则结合，即 $y=(x>1?3:(x<1?2:1))$ 。抓住此知识点，本题就容易了。

12. 以下程序运行结果为（ ）。

```
void main()
{
    int x,y;
    x=5;
    y=6;
    if(x>y)
        printf("%d",x);
    else
        printf("%d",y);
}
```

- A. 5
B. 6
C. 编译出错
D. 值不确定

【提示】通过 if-else 结构判断 x 和 y 值的大小，若 x 大于 y 则输出变量 x 的值，否则输出变量 y 的值。

13. 以下程序运行结果为（ ）。

```
void main()
{
    int x,y;
    x=1;
    y=4;
    switch(x)
    {
        case 0:y--;
        case 1:y++;
```



```
default:y++;x--;  
}  
printf("%d,%d",x,y);  
}
```

A. 0, 5

B. 0, 6

C. 1, 5

D. 1, 6

【提示】在调用 switch 结构时，变量 x 的值为 1，因此执行 y++ 操作。由于没有 break 语句，所以不会退出 switch 结构，会继续执行 y++ 和 x-- 的操作，因此 x 和 y 的最终值为 0 和 6。

14. 以下程序运行结果为 ()。

```
void main()  
{  
    int a,b;  
    a=5;  
    b=8;  
    while(a)  
    {  
        b++;  
        a--;  
    }  
    printf("%d,%d",a,b);  
}
```

A. 0, 13

B. 0, 12

C. 1, 13

D. 1, 12

【提示】在进行 while 循环前，先判断变量 a 的值，若其值不为 0 则执行 b++ 和 a-- 操作，重复执行直到变量 a 的值为 0 为止，退出 while 循环。

15. 以下程序运行结果为 ()。

```
void main()  
{  
    int a,b;  
    a=5;  
    b=4;  
    if(a>b)  
        b++;  
    if(a==b)  
        printf("%d,%d",a,b);  
}
```

A. 没有输出结果

B. 5, 4

C. 5, 5

D. 编译出错

【提示】变量 a 和 b 初始值分别为 5 和 4，变量 a 的值大于 b 的值，因此执行 b++ 操作后，a 和 b 的值都为 5。



二、填空题

1. 以下程序运行结果为_____。

```
void main()
{
    int x=5,y=6;
    int s;
    s=y-x;
    switch(s)
    {
        case 0: printf("first\n");
        case 1: printf("second\n");
    }
}
```

【提示】y-x 的值为 1，因此 s 的值为 1，输出“second”信息。

2. 下列程序的输出结果为_____。

```
#include <stdio.h>
void main()
{
    int a=1,b=2,c=3;
    if(a<b) {a=b;b=c;c=a;}
    printf("%d,%d,%d",a,b,c);
}
```

【提示】变量 a 的值为 1，变量 b 的值为 2，变量 a 的值小于 b，因此将 b 值赋给变量 a，c 值赋给变量 b，a 值赋值给变量 c。

3. 若有以下程序，则输出结果为_____。

```
#include <stdio.h>
void main()
{ int a,b=2;
  if(a=b!=0)
    printf("%d,%d",a,b);
  else
    printf("%d,%d",b,a);
}
```

【提示】a=b!=0 表示先判断 b 是否等于 0，然后将其逻辑值赋值给变量 a，因此该表达式真值始终为 1。

4. 下列程序的运行结果为_____。

```
#include <stdio.h>
void main()
{ int x,y;
  x=(8/3)>2?1:2;
```



```
printf("%d",x);  
}
```

【提示】用条件运算符判断 8 除以 3 的余数，若余数大于 2 则输出 1，否则输出 2。

5. 下列程序的输出结果为_____。

```
#include <stdio.h>  
void main()  
{ int a,b,c,x,y,z;  
  a=1;  
  b=2;  
  c=3;  
  x=(a-1)&&b;  
  y!=(b-2)||a;  
  z=(a+b)&&(b-c);  
  printf("%d,%d,%d",x,y,z);  
}
```

【提示】a 的值为 1，b 的值为 2，c 的值为 3，因此(a-1)&&b 为 0，(a+b)&&(b-c)的值为 1，(b-2)||a 的值为 1。

6. 有一个函数，当 $x > 0$ 时， $y = 2x$ ；当 $x \leq 0$ 时， $y = 4x - 1$ 。通过 scanf() 函数实现 x 值的输入，同时输出 y。通过题意补全下面的空格。

```
#include <stdio.h>  
void main()  
{ int x,y;  
  scanf("%d",____);  
  if(____) y=2*x;  
  else      y=4*x-1;  
  printf("____",____);  
}
```

【提示】本题中调用 scanf() 函数从键盘获取数据，因此其参数应为 &x。然后判断 x 是否大于 0，若大于 0 则 $y = 2x$ ，否则 $y = 4x - 1$ ，最后调用 printf() 函数输出变量 y 的值。

7. 输入数字 1~7，分别打印出对应的星期几（星期一到星期天），在空格中填写完整的代码。

```
#include <stdio.h>  
void main()  
{ int a;  
  scanf("%d",&a);  
  if(____)  
  { switch(____)  
    { case 1:printf("星期一");break;  
      case 2:printf("星期二");break;  
      case 3:printf("星期三");break;  
      case 4:printf("星期四");break;  
    }  
  }  
}
```



```
_____:printf("星期五");break;
case 6:printf("星期六");break;
case 7:printf("星期天");break;
}
____ printf("输入有误");
}
}
```

【提示】因为一个星期只包含星期一到星期天，因此首先判断变量 *a* 的值是否在 1~7 范围内，然后进入 switch 结构输出相应结果，否则输出“输入有误”信息。

8. 以下程序执行的结果为_____。

```
#include <stdio.h>
void main()
{ int x=10,y=20,t=30;
  if(x==y) t=x;x=y;y=t;
  printf("%d,%d",x,y);
}
```

【提示】初学者可能会误以为本题 if 语句后的三条语句都不会执行，其实不然。本题中 *x* 不等于 *y*，因此 *t=x* 不会执行，但其后面的两句仍然会执行。

9. 下面程序运行的结果为_____。

```
#include <stdio.h>
void main()
{ int a=2,b=0;
  switch(a)
  { case 2:switch(b)
    {case 0:printf("zero");break;
    case 1:printf("first");break;
    }
  case 5:printf("five");
  }
}
```

【提示】本题为嵌套 switch 结构范例，变量 *a* 的初始值为 2，因此执行 case 2 语句，进入嵌套 switch 结构。变量 *b* 的初始值为 0，输出 zero，break 语句跳出该结构。然后继续执行外层的 switch 结果中的 case 5，输出 five 信息。

10. 下列程序运行后 *d* 的值为_____。

```
#include <stdio.h>
void main()
{
  int a=3,b=4,c=5,d;
  d=(a<b>c);
  printf("%d",d);
}
```



【提示】表达式 $a < b > c$ 是按照从右向左计算的，先计算 $b > c$ 的值，其值为 0，然后计算 $a < 0$ 的值，其值为 0。

11. 以下程序运行后 t 的值为 8，则输入时 t 的值为_____。

```
#include <stdio.h>
void main()
{ int t;
  scanf("%d",&t);
  if(t>10)    t++;
  else if(t>5) t--;
  else      t=t+5;
}
```

【提示】若输入时 t 的值大于 10 则执行 $t++$ 语句；若 t 的值大于 5 小于 10 则执行 $t--$ 语句；若 t 的值小于 5 则执行 $t=t+5$ 语句，因此要使最终 t 值为 8，输入时 t 值可为 9 或 3。

12. 下列程序运行的结果是_____。

```
#include <stdio.h>
void main()
{ int a=0,b=1,c=0;
  ((a++||c++)&&b++);
  printf("%d,%d,%d",a,b,c);
}
```

【提示】表达式 $((a++||c++)\&\&b++)$ 先计算 $b++$ 的值，然后计算 $a++$ ，再计算 $c++$ 的值。

13. 以下程序的运行结果为_____。

```
void main()
{
  int x=4;
  if(x++>5)
    printf("%d",x--);
  else
    printf("%d",x++);
}
```

【提示】变量 x 初始值为 4，若 $x++$ 大于 5 则输出 $x--$ 的值，否则输出 $x++$ 的值。

14. 以下程序运行结果为_____。

```
void main()
{
  int x=7,y=6,z=5;
  if(x<y>z)
    printf("%d",x);
  else if(y<x>z)
    printf("%d",y);
  else
```



```
printf("%d",z);  
}
```

【提示】表达式 $x < y > z$ 和 $y < x < z$ 的真值都为 0，因此输出变量 z 的值。

15. 以下程序运行输入 5 6 后，则其输出结果为_____。

```
void main()  
{  
    int x,y,s=0;  
    scanf("%d%d",&x,&y);  
    if(x<y)  
        s=s+x+y;  
    else  
        s=s-x-y;  
    printf("%d",s);  
}
```

【提示】从键盘输入 5 6，因此变量 x 和 y 的值为 5 和 6， x 的值小于 y 。因此 $s=s+x+y$ ，变量 s 最终结果为 11。

16. 以下程序运行结果为_____。

```
void main()  
{  
    int x=10,a=1,b=2,c=0,d=1;  
    if(a<b)  
        if(b!=4)  
            if(!c) x=5;  
    else  
        if(d) x=6;  
    printf("%d\n",x);  
}
```

【提示】本题中只要发现最后一个 `if` 条件表达式值为真，则变量 x 赋值为 6 就很简单了，因此可不用管前面的语句。

17. 以下程序运行结果为_____。

```
void main()  
{  
    int x=5,y=6;  
    if(x=y!=0)  
        printf("%d\n",x);  
    else  
        printf("%d\n",x+4);  
}
```

【提示】 $x=y!=0$ 是先执行 $y!=0$ 操作，其值为 1，因此将 1 赋值给变量 x ，该表达式的逻辑值为真。



18. 以下程序运行结果为_____。

```
void main()
{
    int a=5,b=4,c=3;
    if(a=b+c)
        printf("%d",a);
    else
        printf("%d",b+c);
}
```

【提示】 $a=b+c$ 是将 $b+c$ 的值赋值给变量 a ，其逻辑值为真。

19. 以下程序运行结果为_____。

```
void main()
{
    int x,y,s=0;
    x=6;
    y=8;
    ++x;
    s=(++x)+y;
    if(x>y)
        printf("***%d\n",s);
    else
        printf("$${%d\n",s);
}
```

【提示】 $++x$ 后 x 的值变为 7，然后执行 $s=(++x)+y$ ， s 的值为 16， x 的值变为 8。变量 x 的值不大于 y ，因此输出 $$$16$ 。

三、编程题

1. 输入三个数，输出其中的最大数和最小数。

【提示】该题目既可以用 if-else 结构比较，也可以直接比较三个数的大小。

【核心代码】

```
scanf("%d%d%d",&a,&b,&c);
if(a>b&&a>c) ...
if(b>a&&b>c) ...
if(c>a&&c>b) ...
... ..
```

2. 用*号输出字母 C 的图案。

【提示】可以先用*号在纸上写出字母 C，再用函数进行分行输出。

【核心代码】

```
printf(" ***\n");
printf(" *\n");
```




由浅入深学 C 语言——基础、进阶与必做 430 题

```
printf(" * \n");
printf(" ****\n");
```

3. 某班进行学习成绩评估,若学习成绩大于或等于 90 分的同学用 A 表示,60~89 分之间的同学用 B 表示,60 分以下的同学用 C 表示。利用条件运算符的嵌套来完成此题。

【提示】该题可用 if 语句来判断,也可用三目运算符(a>b)?a:b。

【核心代码】

```
g=score>=90?'A':(score>=60?'B':'C');
...
```

4. 输入一个年份,判断该年是否为闰年。

【提示】闰年要进行判断,看其是否能被 100 整除又能被 400 整除,或其能被 4 整除但不能被 100 整除。

【核心代码】

```
scanf("%d",&year);
if(year%400==0||(year%4==0&&year%100!=0))
...
```

5. 输入某年某月某日,判断并输出这一天是这一年的第几天。

【提示】以 3 月 8 日为例,应该先把前两个月的日期加起来,然后再加上 8 天即为本年的第几天。特殊情况闰年且输入月份大于 3 时,需考虑多加一天。

【核心代码】

```
scanf("%d,%d,%d",&year,&month,&day);
switch(month)/*先计算某月以前月份的总天数*/ {
case 1:sum=0;break;
case 2:sum=31;break;
case 3:sum=59;break;
case 4:sum=90;break;
case 5:sum=120;break;
case 6:sum=151;break;
case 7:sum=181;break;
case 8:sum=212;break;
case 9:sum=243;break;
case 10:sum=273;break;
case 11:sum=304;break;
case 12:sum=334;break;
default:printf("data error");break;
}
sum=sum+day; /*再加上某天的天数*/
if(year%400==0||(year%4==0&&year%100!=0))/*判断是不是闰年*/ flag=1;
else
    flag=0;
```



```
if(flag==1&&month>2)/*如果是闰年且月份大于2, 总天数应该加一天*/ sum++;  
...
```

6. 输入一个数, 判断其为奇数还是偶数。

【提示】通过判断整除 2 的余数来判断其是奇数还是偶数。

【核心代码】

```
if(a%2==0) printf(...);  
else      printf(...);
```

7. 从键盘输入两个数, 调换这两个数的值。

【提示】通过另外定义一个变量来保存其中的一个数, 再进行交换。

【核心代码】

```
t=a;a=b;b=t;
```

8. 输入一个不超过 5 位的数字, 求: (1) 求其是几位数; (2) 分别输出这几位数字; (3) 把这几位数分别加 1 再输出。

【提示】首先定义一个长整型变量保存数字, 因为 5 位数很有可能超过整型的最大界限。判断一个不超过 5 位的数是几位数, 可以用该数和一定的数字比较得出。将其整除 10000 即可得到万位上的数字, 同理可得千位、百位、十位、个位上的数字。

【核心代码】

```
long x;...  
if(x>99999) ...  
if(x>9999)  ...  
if(x>999)   ...  
if(x>99)    ...  
if(x>9)     ...  
else       ...  
y=x/10000; ...
```

9. 编写一个程序输入一个学生三次的成绩, 并判断该学生三次是否都及格。若不及格则输出“有不及格科目”, 若都及格则输出三次的平均成绩。

【提示】定义三个浮点型变量保存分数, 用 scanf() 函数进行分数的输入, 并把三次分数和 60 进行比较, 判断是否及格, 若不及格进行相应的输出, 若及格则计算平均分并输出。

【核心代码】

```
float s1,s2,s3;  
scanf("%f%f%f",&s1,&s2,&s3);  
if(s1>60&&s2>60&&s3>60) ...  
else...
```

10. 输出特殊的图案, 把下面代码在 C 环境中运行看看, Very Beautiful!

【提示】字符共有 256 个, 不同的字符显示的图形不一样, 此题就是利用了此原理。

【核心代码】



```
char a=176,b=219;
printf("%c%c%c%c%c\n",b,a,a,a,b);
printf("%c%c%c%c%c\n",a,b,a,b,a);
printf("%c%c%c%c%c\n",a,a,b,a,a);
printf("%c%c%c%c%c\n",a,b,a,b,a);
printf("%c%c%c%c%c\n",b,a,a,a,b);
```

第 6 章 循环结构程序设计

在人们的日常生活中，经常要做重复的事，例如计算 1~100 的累加和，根据前面的知识，可以用 $1+2+3+\cdots+100$ 来求和，但这会显得很复杂。换个角度，可以先令累加器 s 为 0，利用 $s=s+n$ (n 从 1 开始直到 100，一直重复执行此语句)，这种结构就称为循环结构。C 语言中提供了三种循环结构，分别为 for 循环结构、while 循环结构、do-while 循环结构。在本章中，将会讲述这些循环结构及其应用。

本章主要涉及的知识点有：

- 结构化程序设计思想；
- 循环结构程序设计；
- for 循环；
- while 循环；
- do-while 循环；
- 循环的嵌套。



6.1 for 循环

for 循环一般应用在循环次数已知的情況，如求一个数 n 的阶乘，可知总共应进行 $n-1$ 次循环，用 for 循环则很容易实现，对于事先难以确定循环次数的循环用此语句有时不方便。但是若读者认真体会三种循环结构，则会发现，三种循环其实是可通用的，任何一种循环均可用另两种循环实现。

6.1.1 for 循环

for 循环在 C 语言中有着很高的灵活性和效率，其形式如下：

```
for(表达式 1;表达式 2;表达式 3) 循环体语句
```

其中表达式 1 为初始化，一般为一个赋值语句，建立控制循环过程的变量。表达式 2 为循环的条件，条件为一个关系表达式，用来判断什么时候退出循环。表达式 3 为增量，增量用来控制循环过程的变量按照一定的规律变化。

for 语句中的 3 个表达式要用分号隔开，该循环语句中只要表达式 2 条件满足，则继续执行，否则退出 for 循环。其中 for 循环中的循环体语句可以是简单语句，也可以是复合语句。

例如：下面利用 for 循环实现从 1~100 的累加。

```
void main()  
{  
    int s=0,i;
```



```
for(i=1;i<=100;i++)
s=s+i;
printf("%d",s);
}
```

for 循环的执行步骤如图 6-1 所示。

- (1) 计算表达式 1。
- (2) 计算表达式 2，若值为逻辑真，则执行 for 循环中的语句，然后执行第 3 步，若值为逻辑假，则退出 for 循环，执行第 5 步。
- (3) 计算表达式 3。
- (4) 回到第 2 步，继续重新执行。
- (5) for 循环结束，执行其后面的语句。

【范例 6.1】输出 0~50 之间的偶数。

分析：利用 for 循环控制循环变量从 0 变化到 50，判断其奇偶性进行相应的输出。

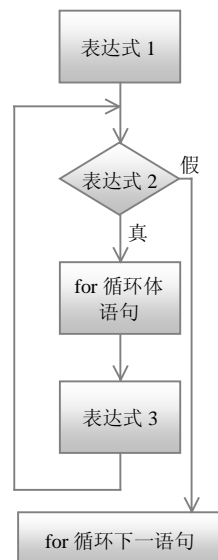


图 6-1 for 循环结构执行过程

范例 6.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int i;
05      for(i=0;i<=50;i++)                               /*for 循环结构*/
06      {
07          if(i%2==0)                                    /*判断 i 除以 2 的余数是否等于 0*/
08              printf("%d ",i);                          /*若等于 0 则输出该变量*/
09          if(!(i%10))                                    /*若该数能被 10 整除*/
10              printf("\n");                              /*换行*/
11      }
12  }
```

【代码分析】本例是 for 循环的简单应用，详细代码分析如下：

- 第 5~11 行为 for 循环结构，用来输出 0~50 之间的偶数。
- 第 7、8 行根据数字除以 2 的余数是否为 0 判断其是否为偶数，若为偶数则输出。
- 第 9、10 行，若 i 能被 10 整除则换行。

【运行结果】该程序的执行结果如图 6-2 所示。

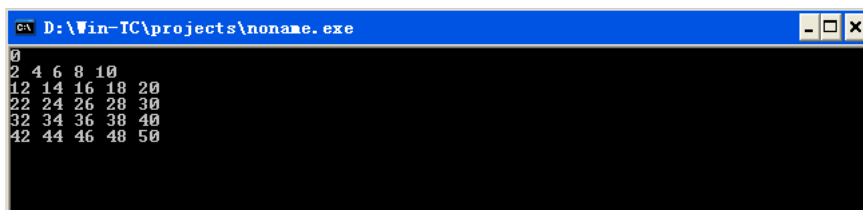


图 6-2 范例 6.1 结果图



6.1.2 for 循环结构应用

for 循环的使用比一般语句要简单、方便得多，例如：

```
void main()
{
    int i;
    for(i=1;i<=50;i++)
        printf("%d\n",i);
}
```

在上面程序中用 for 语句可以轻易地实现 1~50 的输出，若运用其他语句来输出则很麻烦。在 for 循环使用的过程中，可以省去其中的一些表达式。

(1) for 循环中的表达式 1 可以省略，但表达式 1 后的分号不能去掉，而且在调用 for 循环之前就要给循环变量赋值。例如：

```
i=0;
for( ;i<10;i++)
    s=s+i;
```

(2) for 循环中的表达式 2 可以省略，即不判断是否满足循环的条件，for 循环可无限循环下去，一直为真。此时 for 循环为一个死循环，应在内部加循环退出的条件。例如：

```
for(i=0; ;i++)
{
    s=s+i;
    if(i>10) break;
}
```

【范例 6.2】随机抽取两个整数进行整数加法运算，用户输入运算的答案，用户输入答案正确则显示“you are right”，错误则显示“you are wrong”。每次回答问题之后，询问是否继续回答问题，用户输入“n”表示退出，否则继续回答问题。

分析：随机抽取整数需用到 rand() 函数，回答问题的操作可写在 for 循环语句中，若满足则继续执行，否则退出循环。

范例 6.2 代码实现

```
01  #include <stdio.h>
02  #include <stdlib.h>                /*包含 stdlib.h 头文件*/
03  #include <conio.h>                 /*包含 conio.h 头文件*/
04  void main()
05  {
06      int i,x,y,z;
07      char c=' ';                    /*定义一个字符变量，初始值为空*/
08      for(i=1;c!= '\n';i++)
09      {
10          x=rand();                  /*调用 rand() 函数，随机获取数字赋给变量 x*/
```



```
11     y=rand();
12     printf("%d + %d=?\n",x,y);          /*输出要进行的运算操作*/
13     scanf("%d",&z);                      /*从键盘获取用户输入的答案*/
14     if(z==x+y)                          /*若答案正确*/
15         printf("you are right\n");      /*输出 you are right 信息*/
16     else
17         printf("you are wrong\n");
18     printf("Do you want more?\n");      /*提示用户是否需要继续*/
19     c=getche();                          /*获取用户从键盘输入的字符*/
20 }
21 }
```

【代码分析】本例是 for 循环的简单应用，详细代码分析如下：

- 第 1~3 行，包含了三个头文件，分别是 stdio.h、stdlib.h 和 conio.h 头文件。stdio.h 头文件中包含了一些基本输入输出函数，如 scanf() 函数，printf() 函数等。stdlib.h 头文件中包含了一些数学函数，如 abs()、gets() 等函数。conio 头文件中包含了控制数据输入输出的函数，如本题中使用的 getche() 函数。
- 第 10、11 行，rand() 函数为 C 语言提供的一个随机抽取数字的库函数，可以实现随机抽取数据。
- 第 13 行，利用 scanf() 函数输入用户的答案，从而判断其是否正确。
- 第 14~17 行，判断用户输入的答案是否正确，若正确则输出 “you are right”，若不正确则输出 “you are wrong”。
- 第 19 行，利用 getche() 函数获取从键盘输入的字母，用来与 “\n” 比较，若不同则继续执行 for 循环，若相同则退出 for 循环。

【运行结果】该程序的执行结果如图 6-3 所示。

```
C:\ D:\Win-TC\projects\noname.exe
346 + 130=?
476
you are right
Do you want more?
y
10982 + 1090=?
10545
you are wrong
Do you want more?
-
```

图 6-3 范例 6.2 结果图



注意：第 8 行的 for(i=1;c!='\n';i++) 中 i 为无关项，可简化成 for(;c!= '\n';)，使程序更简洁。第 19 行的 getche() 函数为 conio 头文件中的库函数，其功能是从键盘输入字符后立即从控制台取字符，不用以回车为结束。

(3) for 循环中的表达式 3 也可以省略，但分号不能省略。此时应在循环体内加另外的循环变量控制语句，保证 for 循环能够结束，而不是死循环。例如：

```
for(i=0;i<10; )
{
    s=s+i;
```



```
i++;  
}
```

(4) for 循环中表达式 1 和表达式 3 可以同时省略, 只有表达式 2, 即只有循环的条件。此时在 for 循环之前应该循环变量赋初始值, 在 for 循环内部加上循环增量控制语句。例如:

```
int i=0;  
for( ;i<10; )  
{  
    s=s+i;  
    i++;  
}
```

(5) for 循环的三个表达式都可以省去, 但其分号不能省去。例如:

```
for( ; ; )
```

表示没有循环变量, 没有循环条件, 没有循环增量, 即 for 循环中的循环体语句无限执行, 是一个死循环。

【范例 6.3】 利用 for (;;) 循环实现从 1~100 的输出。

分析: 利用 for (;;) 循环, 变量 i 每次加 1 从 1 变化到 100 并输出结果, 当 i 超过范围时, 用 break 语句跳出 for 循环即可。

范例 6.3 代码实现

```
01  #include <stdio.h>  
02  void main()  
03  {  
04      int i=0;  
05      for( ; ; )  
06      {  
07          printf("%d ", ++i);          /*输出++i 的值*/  
08          if(i%10==0)  
09              printf("\n");  
10          if(i==100)                    /*若 i 的值为 100*/  
11              break;                    /*跳出 for 循环*/  
12      }  
13  }
```

【代码分析】 本例是 for 循环的简单应用, 详细代码分析如下:

- 第 10、11 行, 若 i 的值为 100 则跳出 for 循环。

【运行结果】 该程序的执行结果如图 6-4 所示。

(6) for 循环中的表达式 1 和表达式 3 可以为符合 C 语言语法的任意表达式; 可以是循环变量的初始化表达式; 也可以是与循环变量无关的表达式。例如:



```
for(i=0,s=0 ;i<10;i++,s=s+i )
{
    ...
}
```

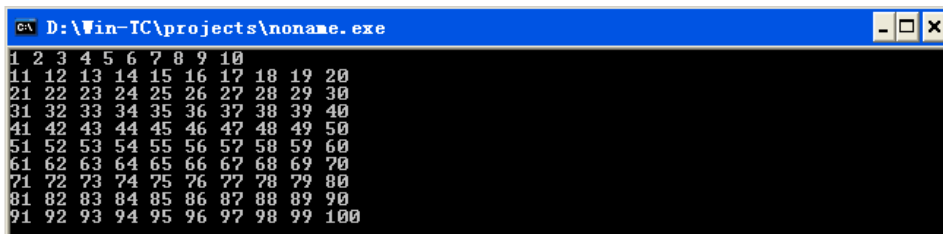


图 6-4 范例 6.3 结果图

上述程序中表达式 1 和表达式 3 为逗号表达式，表达式 1 在给循环变量 *i* 赋 0 的同时也给累加器 *s* 的值赋 0。表达式 3 中 *i* 加 1 后执行 *s=s+i* 表达式。

(7) for 循环中的表达式 2 可以为关系表达式，也可为逻辑表达式，还可为数字表达式或字符表达式，只要表达式 2 真值为 1 则执行 for 循环。例如：

```
for( ;(s=getchar())!='\n'; )
printf("%c",s);
```

上例代码中只有表达式 2，而且为字符表达式，其作用是每输入一个字符后输出该字符，直到遇到“\n”符，跳出 for 循环。



注意：for 循环表达式中虽然允许出现一些与循环变量无关的操作，但这样往往会使程序变得混乱，降低 C 语言程序的可读性。因此建议初学者不要把与循环变量无关的操作放在 for 循环表达式中，仅写一些控制循环变量的操作。



6.2 while 循环

while 循环为 C 语言的另外一种循环结构，其形式如下：

```
while (条件表达式) 语句
```

其中 while 是 C 语言的一个关键字，while 循环中的条件表达式可为符合 C 语言语法的任意表达式，只要该条件表达式的真值为 1，就执行 while 循环中的语句。While 循环可为简单语句，也可为复合语句，复合语句要用“{”和“}”括起来。

while 循环执行的步骤如下：

- (1) 计算条件表达式的真值，若值为 1，则执行步骤 (2)，否则执行步骤 (4)。
- (2) 执行 while 循环中的语句。
- (3) 重新执行步骤 (1)。
- (4) 退出 while 循环结构。

其流程图如图 6-5 所示。

在 while 循环结构的使用过程中应注意以下三个方面：

(1) while 循环体语句若为复合语句，则应用花括号括起来。

(2) while 循环先判断条件表达式的真值，然后执行循环体语句。

(3) 在 while 循环结构中应有循环结束的语句，可以退出 while 循环，否则程序会陷入死循环。

【范例 6.4】 利用 while 循环编写一个程序，实现从键盘输入一个数 n ，求其阶乘即 $n!$ 。

分析：计算一个数的阶乘利用公式 $n!=1\times2\times3\cdots(n-1)\times n$ ，可以利用 while 循环计算其乘积直到乘到 n 为止。

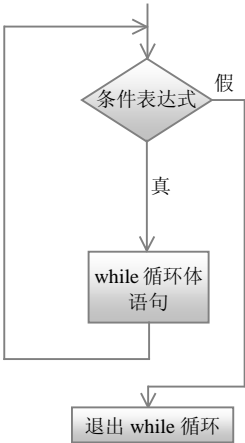


图 6-5 while 循环执行过程流程图

范例 6.4 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x=1,y=1,n;
05      printf("input the number: ");
06      scanf("%d",&n);           /*调用 scanf() 函数实现数据的输入*/
07      while(x<=n)               /*while 循环，其循环条件为 x<=n*/
08      {
09          y=y*x;                 /*将 y*x 的值赋给变量 y*/
10          x++;
11      }
12      printf("the last number is %d\n",y);
13  }
```

【代码分析】 本例是 while 循环的简单应用，详细代码分析如下：

- 第 7~11 行为 while 循环结构，用来实现求 n 的阶乘。

【运行结果】 该程序的执行结果如图 6-6 所示。

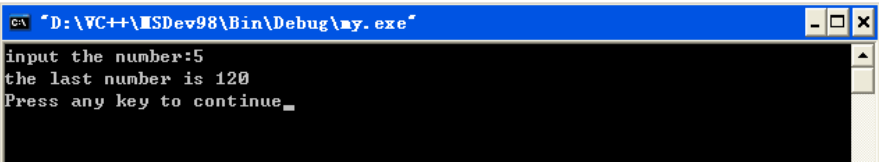


图 6-6 范例 6.4 结果图

【范例 6.5】 利用 while 循环输出计算 1~100 的累加和并输出结果。

分析：while 循环控制变量从 1 加至 100，超过 100 则退出 while 循环并输出结果。

范例 6.5 代码实现

```
01  #include <stdio.h>
02  void main()
```



```
03  {
04      int i=1,s=0;
05      while(i<=100)
06      {
07          s=s+i;
08          i++;
09      }
10      printf("the sum is %d\n",s);          /*输出最后累加的结果*/
11  }
```

【代码分析】本例利用 while 循环求 1~100 的累加和，详细代码分析如下：

- 第 4 行，定义变量 i 和 s，初始化 i 的值为 1。
- 第 5~9 行，通过 while 循环结构实现 1~100 的累加。

【运行结果】该程序的执行结果如图 6-7 所示。

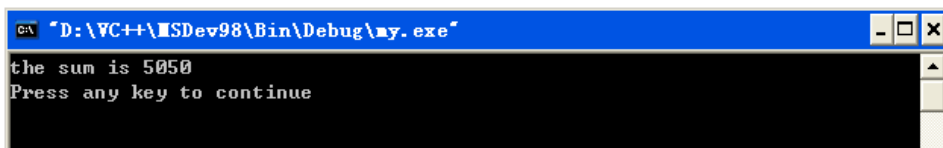


图 6-7 范例 6.5 结果图



6.3 do-while 循环

do-while 循环与 for 循环和 while 循环不同，它不是在头部检验条件，而是在尾部检验。do-while 循环至少执行一次，其形式如下：

```
do
{
    循环体语句
}while(条件表达式);
```

do-while 循环是从 do 开始执行的，先执行一次循环体语句，然后判断 while 中条件表达式的真值，若为真则继续执行循环，否则退出 do-while 循环。

do-while 循环的执行步骤如下：

- (1) 执行 do 后的循环体语句，即先执行一次 do-while 结构中的循环体语句。
- (2) 判断 while 括号中的条件表达式的真值，若为真则执行步骤 (1)，若为假，则执行步骤 (3)。
- (3) 跳出 do-while 循环。

其流程图如图 6-8 所示。

在使用 do-while 循环结构时应注意以下 4 个方面：

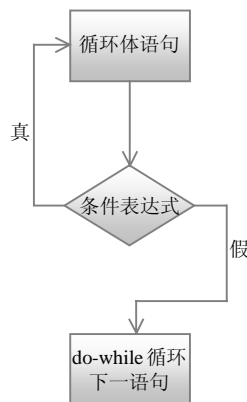


图 6-8 do-while 循环执行过程流程图



(1) do 和 while 都为 C 语言中的关键字，不可改变。

(2) do-while 循环执行由 do 开始，至少执行一次，再判断 while 中条件的真值来决定 do-while 循环是否要继续执行。

(3) while 循环条件表达式后面必须要有个分号，不可丢弃。

(4) 若 do-while 循环中的语句为复合语句，应记得用“{”和“}”括起来。

【范例 6.6】从键盘输入一个数，利用 do-while 循环求其从 1 到该数的累加和。

分析：利用 scanf() 函数及 do-while 循环实现该数字的累加。

范例 6.6 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i=1,n,s=0;
05      printf("input the number: ");
06      scanf("%d",&n);
07      do                                /*do-while 循环*/
08      {
09          s=s+i;
10          i++;
11      }while(i<=n);                    /*若 i>n 则退出 do-while 循环*/
12      printf("the sum is %d\n",s);
13  }
```

【代码分析】本例利用 do-while 循环求累加和，详细代码分析如下：

- 第 7~11 行，通过 do-while 循环实现数字的累积，其中 while 括号条件后的分号要记得加上，否则程序会出错。

【运行结果】该程序的执行结果如图 6-9 所示。

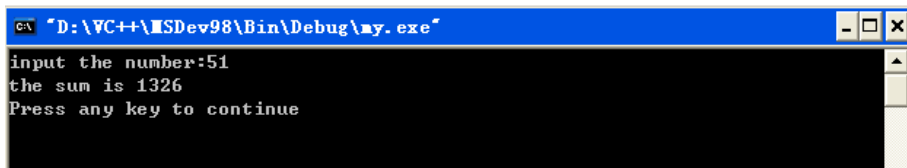


图 6-9 范例 6.6 结果图

【范例 6.7】利用 do-while 语句设计一个简单的菜单。

分析：利用 do-while 构成循环，switch 结构实现简单的菜单功能。

范例 6.7 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int num;
05      do                                /*do-while 循环*/
```



```
06     {
07         printf("*****\n");
08         printf("MENU:\n");           /*输出 MENU 并换行*/
09         printf("1: operation 1\n");
10         printf("2: operation 2\n");
11         printf("3: operation 3\n");
12         printf("4: operation 4\n");
13         printf("*****\n");
14         printf("input the choice number: ");
15         scanf("%d",&num);
16     }while(num<1||num>4);           /*while 循环执行条件num<1 或num大于4*/
17     switch(num)                     /*switch 结构*/
18     {
19         case 1:                     /*case 事件*/
20             printf("you chioce is 1: operation 1");break;
21         case 2:
22             printf("you chioce is 2: operation 2");break;
23         case 3:
24             printf("you chioce is 3: operation 3");break;
25         case 4:
26             printf("you chioce is 4: operation 4");break;
27     }
28 }
```

【代码分析】本例利用 do-while 循环实现了菜单的选择功能，详细代码分析如下：

- 第 5~16 行，利用 do-while 结构输出菜单的界面，以及完成选择的输入。若输入的选择号小于 1 或大于 4 则重新执行 do-while 循环继续输入，直到输入正确结果为止。
- 第 17~27 行是 switch 结构。在该结构中，将输入的选择号与 case 事件比较，若相同则输出相应的语句，并跳出 switch 结构。

【运行结果】该程序的执行结果如图 6-10 所示。

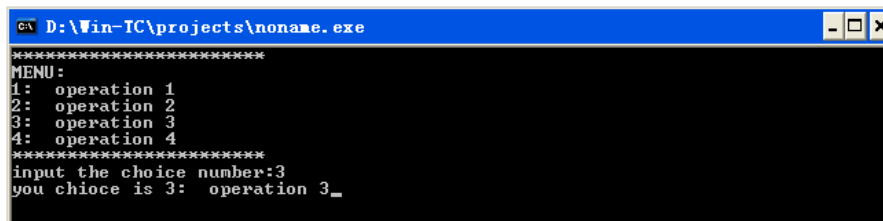


图 6-10 范例 6.7 结果图



6.4 三种循环结构的区别

C 语言中循环结构包括 for 循环结构、while 循环结构、do-while 循环结构。下面主要讲解这三种循环结构的区别。



for 循环的形式一般如下:

```
for(初始化;条件;增量)
{语句}
```

其中 for 循环通过条件可以控制语句的执行次数。

while 循环的形式一般如下:

```
while(条件)
{语句}
```

其中 while 循环条件如满足则执行语句, 不满足则退出 while 循环。

do-while 循环的形式一般如下:

```
do
{语句}
while(条件);
```

其中 do-while 循环中的语句至少会执行一次, 即从 do 开始执行再判断 while 中的条件真值来决定 do-while 循环是否需要执行。

while 循环和 do-while 循环通常用于循环次数不定的情况, 而 for 循环一般用于循环次数有限的情况。这 3 种循环处理同一问题时, 一般都可以通用。例如: 求 5 个数中的最大者。

若用 for 循环求解, 如下所示。

```
void main()
{
    int x,i,max;
    scanf("%d",&x);
    max=x;
    for(i=1;i<5;i++)
    {
        scanf("%d",&x);
        if(max<x)
            max=x;
    }
    printf("the max is %d",max);
}
```

运行结果如图 6-11 所示。

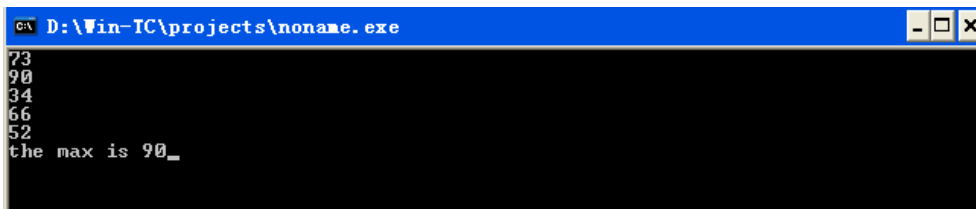


图 6-11 for 循环示例结果图

若用 while 循环可改为:



```
void main()
{
    int x,i,max;
    scanf("%d",&x);
    max=x;
    i=2;
    while(i<=5)
    {
        scanf("%d",&x);
        if(max<x)
            max=x;
        i++;
    }
    printf("the max is %d",max);
}
```

用 do-while 循环语句可改写如下：

```
void main()
{
    int x,i,max;
    scanf("%d",&x);
    max=x;
    i=2;
    do
    {
        scanf("%d",&x);
        if(max<x)
            max=x;
        i++;
    }while(i<=5);
    printf("the max is %d",max);
}
```

这三种循环结构虽然可以通用，但应根据具体情况选择不同的循环结构，可使程序更加简单。如循环次数不定，一般采用 while 循环或 do-while 循环，第一次必然发生的则用 do-while 循环。



6.5 嵌套循环

嵌套循环是指一个循环包含另外一个循环，这种结构也称为多重循环。嵌套循环的形式主要包括以下 4 种：

1. while 循环嵌套

```
while()
{
```



```
...
while()
{
    ...
}
```

2. do-while 循环嵌套

```
do
{
    ...
    do
    {
        ...
    } while();
} while();
```

3. for 循环嵌套

```
for( ; ; )
{
    ...
    for( ; ; )
    {
        ...
    }
}
```

4. for 循环与 while 循环的嵌套

```
for( ; ; )
{
    ...
    while()
    {
        ...
    }
}
```

【范例 6.8】 利用循环嵌套，输出 1~10 的 2 倍数、4 倍数、6 倍数。

分析：用多重循环，外层循环变量控制从 1 变化至 10，内层循环控制按倍数的规律变化，计算后输出即可。

范例 6.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i,j,k;
```




```
05     printf(" i    2i    4i    6i\n");
06     for(i=1;i<=10;i++)
07     { printf("%4d",i);                      /*以%4d 格式输出变量 i 的值*/
08       for(j=2;j<=6;j=j+2)
09       {
10         k=i*j;                          /*将 i*j 的值赋值给变量 k*/
11         printf("%4d",k);                  /*调用 printf()函数输出变量 k 的值*/
12       }
13     printf("\n");
14   }
15 }
```

【代码分析】本例是嵌套循环的简单范例，详细代码分析如下：

- 第 6~14 行是嵌套循环，利用其进行数据的相乘并将结果进行分行输出。
- 第 8~12 行是内部嵌套循环，第 10 行计算 i 和 j 的乘积，并将结果输出，其中“%4d”表示输出的字段宽度为 4。

【运行结果】该程序的执行结果如图 6-12 所示。

i	2i	4i	6i
1	2	4	6
2	4	8	12
3	6	12	18
4	8	16	24
5	10	20	30
6	12	24	36
7	14	28	42
8	16	32	48
9	18	36	54
10	20	40	60

图 6-12 范例 6.8 结果图

【范例 6.9】利用嵌套循环，编写程序输出九九乘法表。

分析：可控制嵌套外层变量从 1 变化至 9，内层循环变量也从 1 变化至 9，相乘后输出结果即可。

范例 6.9 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i,j;
05      for(i=0;i<=9;i++)                      /*for 循环外层循环*/
06      {
07          for(j=0;j<=9;j++)                  /*for 循环内层循环*/
08          printf("%d X %d =%-4d",i,j,i*j);    /*输出变量 i 和 j 及其乘积*/
09          printf("\n");
10      }
11  }
```

【代码分析】本题利用嵌套循环输出九九乘法表，详细代码分析如下：



- 第5~10行为for循环嵌套结构，通过循环计算i和j的乘积并输出。



注意：第8行中输出i和j的乘积格式为“%-4d”，不是“%4d”。它表示让i*j的乘积左对齐，即结果靠左边显示，否则会与i在一起显示，很容易混淆。

【运行结果】该程序的执行结果如图6-13所示。

```

0*0=0
1*0=0 1*1=1
2*0=0 2*1=2 2*2=4
3*0=0 3*1=3 3*2=6 3*3=9
4*0=0 4*1=4 4*2=8 4*3=12 4*4=16
5*0=0 5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*0=0 6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*0=0 7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*0=0 8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*0=0 9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
  
```

图 6-13 范例 6.9 结果图



6.6 循环结构强化实例

【范例 6.10】百鸡问题：若一个鸡翁值五钱，一个鸡母值三钱，三个鸡稚值一钱，现有一百钱，要买一百只鸡，应买鸡翁、鸡母、鸡稚各多少只？

分析：利用for循环穷举所有可能的组合，并利用选择结构输出符合要求的组合。其中鸡翁a的取值范围为0~20只，鸡母b的取值范围为0~33只，鸡稚c的取值范围为100-a-b只。

范例 6.10 代码实现

```

01  #include <stdio.h>
02  void main()
03  {
04      int a,b,c;
05      printf(" a b c\n");
06      for(a=0;a<=20;a++)          /*for 循环控制 a 从 0 变化至 20*/
07          for(b=0;b<=33;b++)
08          {
09              c=100-a-b;          /*c 的值为 100-a-b*/
10              if((c%3==0)&&((5*a+3*b+c/3)==100)) /*若该组合价格为 100 元并且 c 是 3 的倍数*/
11                  printf("%3d%3d%3d\n",a,b,c);    /*输出该组合*/
12          }
13  }
  
```

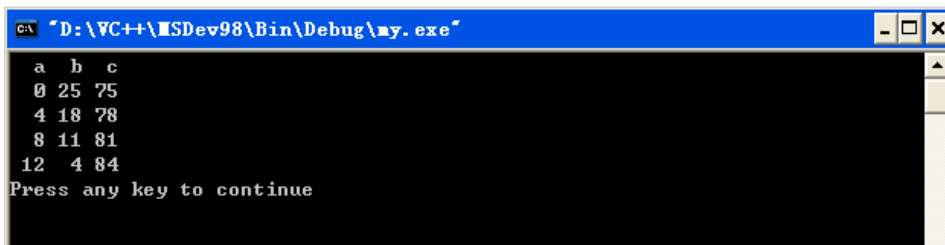
【代码分析】本例利用循环计算百钱百鸡的可能组合，详细代码分析如下：

- 第6~12行是for循环2重嵌套结构，通过穷举把所有可能的组合都进行判断，若符合要求则输出该组合。
- 第10行，鸡稚3个值一钱，因此鸡稚的个数必须为3的倍数即整除3的余数为0，并



判断该组合是否能达到一百钱。

【运行结果】该程序的执行结果如图 6-14 所示。



```

a  b  c
0 25 75
4 18 78
8 11 81
12 4 84
Press any key to continue

```

图 6-14 范例 6.10 效果图

【范例 6.11】编写一个程序，输出 1~100 之间的素数。素数是指除了 1 和其本身之外不能被其他数整除的整数。

分析：判断一个数 n 是否为素数，可以判断该数是否能被 $2 \sim (n-1)$ 整除即可，若都不能被整除则为素数。另外在外层加一个 for 循环，控制变量从 1 变化至 100 即可。

范例 6.11 代码实现

```

01  #include <stdio.h>
02  void main()
03  {
04      int i,j,k=0;
05      for(i=1;i<100;i++)                /*for 循环结构控制变量 i 的变化*/
06      {
07          for(j=2;j<=i-1;j++)
08          {
09              if(i%j==0)                /*若 i 能被 j 整除*/
10                  break;                /*跳出 for 循环*/
11              if(j>=i)
12                  { printf("%4d",i);      /*输出为素数的数字*/
13                    k++;                  /*统计素数个数*/
14                  }
15              if(!(k%10))
16                  printf("\n");
17          }
18      }
19  }
```

【代码分析】本例利用循环穷举求 1~100 之间的素数，详细代码分析如下：

- 第 5~18 行是 for 循环的嵌套，用来穷举 1~100 之间的数并判断是否为素数。
- 第 7~17 行是内层 for 嵌套，用来判断一个数是否为素数，即该数是否能被 $2 \sim (n-1)$ 整除，若能被整除则退出该循环，若不能被整除则为素数，输出该数。

【运行结果】该程序的执行结果如图 6-15 所示。

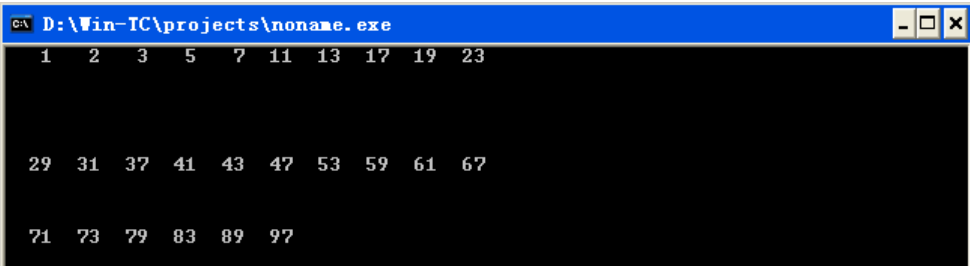


图 6-15 范例 6.11 结果图

【范例 6.12】现有一张 100 元钞票，专门用来购买 3 元、4 元、5 元的商品，求总共有多少种买法可以恰好将 100 元花完。

分析：假设 3 元、4 元、5 元的商品分别有 x 、 y 、 z 件，则 x 的取值范围为 $0\sim33$ ， y 的取值范围为 $0\sim25$ ， z 的取值范围为 $0\sim20$ ，可以利用 for 循环穷举进行判断求解。

范例 6.12 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x,y,z;
05      printf(" x y z");
06      for(x=0;x<=33;x++)          /*通过 for 循环控制变量 x 从 0 变化至 33*/
07          for(y=0;y<=25;y++)
08              for(z=0;z<=20;z++)
09                  if(3*x+4*y+5*z==100)      /*若该组合价格为 100*/
10                      printf("%4d%4d%4d\t",x,y,z);      /*输出该组合并水平移动一段距离*/
11  }
```

【代码分析】本例利用循环穷举求解，详细代码分析如下：

- 第 6~10 行是 3 重 for 循环嵌套，对所有可能的情况进行判断，若满足要求则输出结果。
- 第 9~10 行，判断是否满足条件，即 3 件商品的花销是否为 100 元，若满足该条件则输出商品对应的数目。

【运行结果】该程序的执行结果如图 6-16 所示。

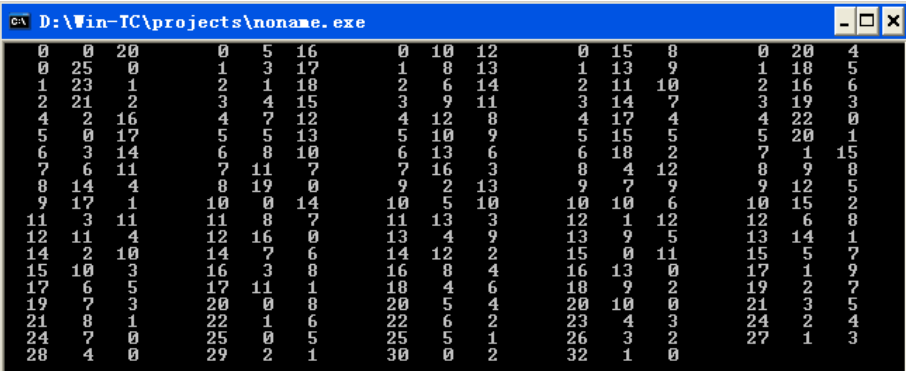


图 6-16 范例 6.12 结果图



【范例 6.13】现有一个 Fibonacci 数列 1, 1, 2, 3, 5, 8……, 求其第 40 个数。

分析: 从上面的数字可以看出 Fibonacci 数列有一个规律, 即每个数等于前两个数之和, 因此可设 $F_1=1$, $F_2=1$, 用公式 $F_n=F_{n-1}+F_{n-2}$ 求解。

范例 6.13 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i,f1=1,f2=1,f=0;
05      for(i=1;i<=38;i++)
06      {
07          f=f1+f2;                /*将 f1 加上 f2 的值赋值给变量 f*/
08          f1=f2;                  /*将 f2 值赋给变量 f1*/
09          f2=f;                    /*将 f 的值赋给变量 f2*/
10      }
11      printf("the 40th number is %d",f);    /*输出运算得出的结果*/
12  }
```

【代码分析】本题利用循环求 Fibonacci 数列, 详细代码分析如下:

- 第 4 行, 定义 4 个整型变量, 其中 i 用于控制循环次数, $f1$ 、 $f2$ 分别表示 Fibonacci 数列的第一个和第二个数, f 用来保存两个数相加的和。
- 第 5~10 行, for 循环执行了 38 次, 即求得第 40 个数。

【运行结果】该程序的执行结果如图 6-17 所示。

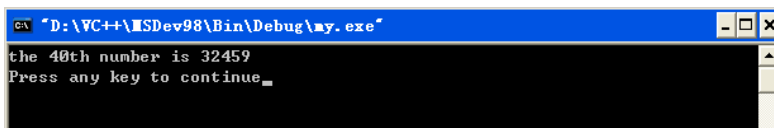


图 6-17 范例 6.13 结果图



6.7 小结

本章讲解了 C 语言中的 3 种循环结构, 分别为 for 循环、while 循环和 do-while 循环。循环结构可以减少程序的书写量, 它包含三个要素: 循环变量、循环体、循环条件。当要重复执行一段代码时, 循环结构可以最简便、最高效地执行该算法。



6.8 习题

一、选择题

1. 以下程序的输出结果正确的是 ()。

```
#include <stdio.h>
```



```
void main()
{
    int i;
    for(i=1;i<=3;i++)
        if(i%2)
            printf("$");
        else
            printf("*");
}
```

A. \$\$*

B. \$\$

C. **

D. **

【提示】本题考察 for 循环的应用，for 循环变量从 1 增至 3，若能被 2 整除则输出 “\$”，否则输出 “*”。

2. 若从键盘输入 abc*，以下程序的输出结果正确的是（ ）。

```
#include <stdio.h>
void main()
{
    int x=0,y=0;
    char c;
    while((c=getchar())!='*')
        switch(c)
        {
            case 'a':x++;
            case 'b':y++;
            case 'd'   ;:
        }
    printf("%d,%d\n",x,y);
}
```

A. 1, 1

B. 1, 2

C. 2, 2

D. 3, 3

【提示】本题在 while 循环中嵌套了 switch 结构，因为 switch 结构中没有 break 语句，程序响应事件执行之后并不会马上退出，而会继续往下执行，直到 switch 结构末尾。当从键盘输入 “abc*” 时，getchar() 函数首先获取第一个字符即 a，进入 switch 结构中 x 和 y 的值都加 1。第 2 个字符 b，y 的值加 1，x 的值不变。第 3 个字符 c，x、y 的值保持不变，因此最后 x、y 的值分别为 1、2。

3. 有以下程序：

```
int i=0;
while(i=0)
    i++;
```

其中 while 循环执行的次数为（ ）。

A. 一次

B. 两次



由浅入深学 C 语言——基础、进阶与必做 430 题

C. 无限次

D. 一次都不执行

【提示】本题初学者很容易选 A 选项，认为 while 循环执行一次。其实 while 循环中的 `i=0` 为赋值语句，并非关系表达式，其真值将一直为 1，因此 while 循环将会无限执行下去。

4. 关于以下程序说法正确的是（ ）。

```
#include <stdio.h>
void main()
{
    int i,j;
    for(i=1,j=3;(j<6)||i>3;i++)
        printf("*");
}
```

A. for 循环执行 3 次

B. for 循环执行无限次

C. for 循环执行次数不定

D. for 循环执行 6 次

【提示】本题考察 for 循环条件的判断，其中 `j<6||i>3` 为 for 循环的条件，j 初始值为 3，因此 `j<6` 的真值为 1，`j<6||i>3` 的真值也为 1，因此 for 循环将会无限执行。

5. 下列程序运行后，b 的值为（ ）。

```
#include <stdio.h>
void main()
{
    int a=2,b=4;
    do
    {
        b=b-a;
        ++a;
        ++b;
    }while(b<0)
}
```

A. 2

B. 3

C. 4

D. 都不正确

【提示】本题考察 do-while 循环，它先执行循环再判断条件。上述程序中，b 先减去 a 值变为 2，再加上 1 变为 3，while 括号中 `b<0` 条件不满足则退出 do-while 循环。

6. 下列程序运行后，c 的值为（ ）。

```
#include <stdio.h>
void main()
{
    int a,b,c=1;
    for(i=0;i<4;i++)
        for(j=5;j>0;j--)
            c++;
}
```

A. 20

B. 21



C. 24

D. 25

【提示】本题为 for 循环的双重嵌套，外层循环变量 a 从 0 变化至 3，内层循环变量 b 从 5 减至 1，总共执行 20 次，同时 c 的初始值为 1，因此 c 的最终值为 21。

7. 以下程序运行结果为 ()。

```
void main()
{
    int s=0,x=5;
    while(x)
    {
        s=s+x;
        x--;
    }
    printf("%d",s);
}
```

A. 15

B. 10

C. 16

D. 11

【提示】通过 while 循环计算 1 到 5 累加和，因此 s 的值为 15。

8. 以下程序中 for 循环执行的次数为 ()。

```
void main()
{
    int x=5;
    for( ;x>0;x--)
        x--;
}
```

A. 2

B. 3

C. 4

D. 5

【提示】本题中 for 循环语句为 x--，表达式也为 x--，因此总共执行次数为 3 次。

9. 以下程序运行结果为 ()。

```
void main()
{
    int x;
    x=3;
    while(x<=5)
    {
        x++;
        printf("%d",x);
    }
}
```

A. 345

B. 456

C. 3456

D. 4567



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】进入 while 循环时，变量 x 初始值为 3，执行 x++，然后输出变量 x 的值，直到 x 的值大于 5 才退出 while 循环。

10. 以下程序运行结果为 ()。

```
void main()
{
    int x;
    for(x=1;x<=10;x++)
        if(x%3==0)
            printf("%d",++x);
}
```

A. 4710

B. 369

C. 47

D. 36912

【提示】1~10 中能被 3 整除的为 369，本题 for 循环中加 1 后再输出其值，因此最终结果为 4710。

二、填空题

1. 下列程序的输出结果为_____。

```
#include <stdio.h>
void main()
{
    int i,j=0;
    for(i=1;i<=7;i++)
        switch(i)
        {
            case 1:
            case 5:j++;
            case 6:j++;break;
            case 7:j++;break;
        }
    printf("%d",j);
}
```

【提示】本题为 for 循环和 switch 结构的嵌套，switch 结构中 case 1 和 case 5 后没有 break 语句，因此 j++ 总共执行次数为 6 次。

2. 下列程序输出 1~100 之间的偶数，补全下面的代码。

```
#include <stdio.h>
void _____()
{
    _____ i;
    for(i=1;i<=100;i++)
        printf("_____",i);
}
```



【提示】若一个数为偶数，则其除以 2 的余数一定为 0。

3. 下列程序的输出结果为_____。

```
#include <stdio.h>
void main()
{
    int x,y;
    for(x=1,y=2;x<=10;x++)
    {
        if(y>6)
            break;
        y=y+2;
    }
    printf("%d,%d",x,y);
}
```

【提示】本题中 for 循环执行条件为 $x \leq 10$ ，同时若 $y > 6$ 则退出 for 循环。

4. 下列程序中 do-while 循环执行的次数是_____，输出的结果为_____。

```
#include <stdio.h>
void main()
{
    int a=4,b=1;
    do
    {
        b++;
        a=a-2;
    }while(a>0);
    printf("a=%d,b=%d",a,b);
}
```

【提示】本题 do-while 结构中先执行 $b++$ 和 $a=a-2$ 语句，然后判断 a 值是否大于 0，若满足则继续执行 do-while 循环，否则退出 do-while 循环。

5. 现有以下程序，其功能是从若干场比赛的分数（0~100）中，输出比赛所得最多分和比赛所得最低分，当输入负数时结束分数的输入，补全下面的代码。

```
#include <stdio.h>
void main()
{
    int x,max,min;
    scanf("%d",&x);
    max=x;
    min=x;
    while(_____)
    {
        if(_____)
            max=x;
    }
```



```
    if(min>x)
        min=x;
}
printf("max=_____,min=_____" ,max,min);
}
```

【提示】输入负数时结束分数的输入，则 while 循环条件应判断分数是否大于 0，然后比较分数，若 max 小于该分数则将该分数赋值给变量 max。

6. 以下程序运行结果为_____。

```
void main()
{
    int x=5,y=0;
    do
    {
        x--;
        if(x%3==0)
            continue;
        y=y+x;
    }while(x>0);
    printf("%d",y);
}
```

【提示】do-while 循环中若 x 能被 3 整除则返回 do-while 循环重新执行，否则 y=y+x，直到 x 的值不大于 0 为止退出 do-while 循环，输出变量 y 的值。

7. 以下程序运行结果为_____。

```
void main()
{
    int x=4,s=1;
    for(;x>0;x--)
    {
        if(x%2==0)
            continue;
        s=s*x;
    }
    printf("%d",s);
}
```

【提示】for 循环执行条件为 x>0，若 x 能被 2 整除则返回 for 循环减 1 继续执行，否则 s=s*x，直到 x 的值不大于 0 为止退出 for 循环。

8. 以下程序的运行结果为_____。

```
void main()
{
    int x,y;
    for(x=5;x>0;x--)
```



```
{
    for(y=x;y>0;y--)
        printf("*");
    printf("\n");
}
```

【提示】本题利用 for 循环嵌套输出相应的图形。

9. 以下程序运行结果为_____。

```
void main()
{
    int x,y;
    x=5;
    switch(x)
    {
        case 5:printf("*");
        case 6:printf("***");break;
        default:printf("****");
    }
    y=x*x+2;
    printf("%d",y);
}
```

【提示】进入 switch 结构中, x 的值为 5, 输出 “*”, 但由于没有 break 语句不会退出 switch 结构, 因此会继续输出 “**”, 最后输出变量 y 的值。

10. 以下程序运行结果为_____。

```
void main()
{
    int x=5,y=6;
    if(x=y!=0)
        printf("%d\n",x);
    else
        printf("%d\n",y);
}
```

【提示】x=y!=0 中先计算 y!=0 的真值, 然后将该真值赋值给变量 x。

11. 以下程序运行结果为_____。

```
void main()
{
    int x,y,z;
    x=6;
    y=5;
    z=(x>y)?x++:++y;
    printf("%d",z);
}
```



【提示】 $(x > y) ? x++ : ++y$ 中 $x > y$ 成立，因此 $z = x++$ 。

12. 以下程序运行结果为_____。

```
void main()
{
    int x=0,y=0,z=0;
    switch(x)
    {
        case 0: switch(y)
        {
            case 0: z++; break;
            case 1: z++; break;
        }
        case 1: z++; y++;
    }
    printf("%d,%d", y, z);
}
```

【提示】本题为 switch 结构的嵌套，在求解的过程中注意 break 语句及变量值的变化即可。

13. 以下程序运行结果为_____。

```
void main()
{
    int x=4,y=5;
    if(x >= --y)
        printf("%d", x);
    else
        printf("%d", y);
}
```

【提示】 $x \geq --y$ 的真值为 1，因此输出变量 x 的值。

14. 以下程序运行结果为_____。

```
void main()
{
    int x=0,y=0;
    int i,j,s=0;
    for(i=0; i<=2; i++)
        for(j=0; j<3; j++)
            s=s+i+j;
    printf("%d", s);
}
```

【提示】本题为 for 循环的嵌套，外层 for 循环变量 i 从 0 变化至 2，内层 for 循环变量 j 从 0 变化至 3，将 $s+i+j$ 的值赋给变量 s。

15. 以下程序运行结果为_____。

```
void main()
```



```
{
    int n;
    float s=0.0;
    for(n=5;n>1;n--);
    s=s+1.0/n;
    printf("%.1f",s);
}
```

【提示】本题通过 for 循环计算 $1/5+1/4+1/3+1/2$ 的值，最后输出精度为小数点后一位。

16. 以下程序运行结果为_____。

```
void main()
{
    int i,s=0;
    for(i=1;i<10;i++)
    {
        if(i%3==0) continue;
        s=s+i;
    }
    printf("%d\n",s);
}
```

【提示】本题通过 for 循环控制 i 从 1 变化至 10，若 i 除以 3 的余数为 0 则返回 for 循环加 1 继续执行，否则 $s=s+i$ ，最后输出变量 s 的值。

17. 以下程序运行结果为_____。

```
void main()
{
    int i,j;
    for(i=1;i<=3;i++)
    {
        for(j=i;j>0;j--)
            printf("#");
        printf("\n");
    }
}
```

【提示】本题通过嵌套 for 循环结构，外层循环控制变量 i 从 1 变化至 3，内层循环控制变量 j 从 i 变化至 1，输出相应的图形。

18. 以下程序运行结果为_____。

```
void main()
{
    int i,s=1;
    for(i=1;i<5;i++)
        s=s*i;
    printf("%d\n",s);
}
```



```
}
```

【提示】本题通过 for 循环计算 5! 的值并输出至屏幕。

三、编程题

1. 编写一个程序，求出 1000 以内的完全数。

【提示】如果一个数等于其因子之和，则该数为完全数，例如 6 的因子为 1, 2, 3, $1+2+3=6$ ，因此 6 为完全数。利用 for 循环列举 1~1000 中所有的数，进行判断，若为完全数则输出。

【核心代码】

```
for(i=1;i<=1000;i++)
{
    for(j=1,s=0;j<=i/2;j++)
        if(i%j==0) s=s+j;
    if(s==i)
        printf("%3d",i);
}
```

2. 百马百担问题：现有 100 匹马，100 担货，大马驮 3 担，中马驮 2 担，两匹小马驮 1 担，问大马、中马、小马应各需多少匹。

【提示】100 担货可以确定大马的范围为 0~33 匹，中马的范围为 0~50 匹，可用 for 循环嵌套来求解。

【核心代码】

```
for(i=0;i<=33;i++)
for(j=0;j<=50;j++)
{
    k=100-i-j;
    if(3*i+2*j+k/2==100)
        printf("%d,%d,%d\n",i,j,k);
}
```

3. 编写程序求出 0~200 中能被 4 整除余 3 的数。

【提示】利用 for 循环变量从 0 变化至 200，对其中的每个数进行判断，若满足条件则输出。

【核心代码】

```
for(i=0;i<200;i++)
    if(i%4==3)
        printf("%d ",i);
```

4. 从键盘输入一个数 n 求其阶乘 n!。

【提示】用 scanf() 函数进行数字的输入，同时利用 for 循环控制循环变量从 1 变化至 (n-1) 进行相乘，输出最后计算的结果即可。

【核心代码】

```
scanf("%d",&n);
for(i=1;i<=n-1;i++)
```



```
s=s*i;
printf("%d",s);
```

5. 现有 100 元, 用来买 7 元、8 元、9 元的商品, 问将 100 元花完总共有多少种方法?

【提示】利用 3 重嵌套 for 循环, 穷举所有可能的情形, 输出符合要求的组合即可。

【核心代码】

```
int x,y,z;
for(x=0;x<=16;x++)
for(y=0;y<=13;y++)
for(z=0;z<=12;z++)
if(7*x+8*y+9*z==100)
printf("%d,%d,%d",x,y,z);
```

6. 猜数游戏: 游戏中随机产生一个 1~100 之间的数, 请用户猜数。若 10 次之内猜中则提示成功, 若 10 次之内没有猜中则提示失败。

【提示】随机产生一个数可以使用 random()函数和 randomize()函数, 它们包含在 stdlib.h 头文件中。利用 for 循环控制进行 10 次猜数, 猜中则提示成功, 没猜中则提示失败。

【核心代码】

```
#include <stdlib.h>
...
i=1;
randomize();
n=random(100)+1;
printf("input a number(1~100)\n");
do
{
...
scanf("%d",&g);
i++;
if(g==n)
printf("sucess\n");
...
else
...
}while(i<=10);
```

7. 求出 100~400 中满足条件的所有数, 要求三个数字之积为 24, 三个数之和为 12。

【提示】利用穷举法求解, 对 100~400 中所有数字进行判断, i、j、k 分别表示个位数、十位数、百位数, 利用 for 循环判断求解。

【核心代码】

```
for(i=1;i<=4;i++)
for(j=1;j<=9;j++)
for(k=1;k<=9;k++)
if(i*j*k==24&& i+j+k==12)
```




```
printf("%d%d%d\n",i,i,k);
```

8. 从键盘输入 20 个数，求出其中正数的个数及其平均值。

【提示】从键盘输入 20 个数，可以通过 for 循环和 scanf() 函数来实现。另外设置一个变量 sum，初始值为 0，用于计算其累加和。

【核心代码】

```
for(i=0;i<20;i++)
    scanf("%d",&x[i]);
for(i=0;i<20;i++)
{
    if(x[i]>0)
        m++;
    sum=sum+x[i];
}
aver=sum/20.0;
```

9. 编写一个程序，打印出所有的水仙花数。

【提示】水仙花数是指一个 3 位数中，其各个位的数值的立方和等于该数。如 $153=1^3+5^3+3^3$ ，因此 153 为水仙花数。

【核心代码】

```
for(i=100;i<=999;i++)
{
    x=i/100;
    y=(i-x*100)/10;
    z=i%10;
    if(x*x*x+y*y*y+z*z*z==i)
        printf("%d\n",i);
}
```

10. 输入一串字符，统计其中的字母、空格、数字及其他字符的个数。

【提示】将键盘输入的一串字符保存在字符数组中，然后设置计数器对其中的字符逐个进行判断统计字母、空格、数字及其他字符的个数。

【核心代码】

```
scanf("%s",s);
while(s[i]!='\0')
{
    if(s[i]>= 'a'&& s[i]<= 'z')
        a++;
    else if(s[i]>= 'A'&& s[i]<= 'Z')
        b++;
    else if(s[i]== ' ')
        c++;
    else
        d++;
}
```



```
i++;  
}
```

11. 编写程序，从键盘输入一个数，判断是否为素数并输出相应的信息至屏幕。

【提示】判断一个数是否为素数，可以通过整除 $2 \sim n-1$ 中的数来判断。若该数不能被 $2 \sim n-1$ 中的数整除，则为素数，否则不是素数。

【核心代码】

```
scanf("%d",&n);  
for(i=2;i<=n/2;i++)  
if(n%i==0)  
flag=0;  
if(flag==1)  
printf("该数为素数\n");  
else  
printf("该数不是素数\n");
```

12. 编写程序，打印九九乘法表。

【提示】打印九九乘法表，可以通过两重 for 循环来实现，外层变量 i 从 0 变化至 9，内层变量 j 从 0 变化至 i 。

【核心代码】

```
for(i=0;i<=9;i++)  
{  
for(j=0;j<=i;j++)  
printf("%dx%d=%-4d");  
printf("\n");  
}
```

第 2 篇 C 语言技术进阶

第 7 章 数 组

在前面的章节中介绍了一些基本的数据类型，但只能解决一些简单问题。一旦遇到一些复杂的问题就很难解决，例如要定义 50 个整型变量，若一个一个地定义，则会很复杂。因此 C 语言中提供了一种结构专门来处理这种问题，即数组。数组是相同类型数据的集合，它们都拥有同一个名称。数组可以是一维数组，也可以为二维数组，若要访问数组中的内容，可通过下标进行访问。本章将介绍数组的概念及其应用。

本章主要涉及的知识点有：

- 数组的定义；
- 多维数组；
- 数组的应用；
- 指针数组；
- 字符串。



7.1 数组简介

在程序设计的过程中，经常会处理一些数据类型相同的变量，为了方便，C 语言中提供了数组这一结构。它把同一类型的数据有序进行排列，是同种类型数据的集合。按照数组元素类型的不同，数组可以分为整型数组、字符型数组、指针数组；按照数组空间大小，又可以分为一维数组、二维数组、多维数组。

数组对同一类型的一系列数据用一个名称，通过下标（数字）来引用数据。使用数组可以简化程序，并且利用循环可使程序的执行效率变高。数组有上界和下界，数组中的元素是存储在上界和下界之间的，并且是连续的。

在 C 语言中，一个数组中的元素，其数据类型必须相同，即为定义数组时的数据类型。数组的一般形式为：类型 数组名[数组大小]，其中的类型可为任一个基本数据类型和构造数据类型。数组名为用户自定义数组的名称，数组大小即数组内元素的个数，也称为数组长度。

例如：

```
int x[100];           /*定义长度为 100 的整型数组 x*/  
float y[10],z[100];   /*定义长度为 10 的单精度浮点型数组 y 和长度为 100 的单精度浮点型数组 z*/  
double a[10],b[100];  /*定义长度为 10 的双精度浮点型数组 a 和长度为 100 的双精度浮点型数组 b*/  
char c[100];          /*定义长度为 100 的字符数组 c*/
```

7.2 为何需要数组

数组可用来解决复杂的问题，在 C 语言中起着至关重要的作用，例如下面这个例子。

【范例 7.1】 现有一个班级的学习成绩表如表 7-1 所示，要求计算每个学生的平均成绩并输出，如表 7-2 所示。

表 7-1 学生成绩表

姓 名	语 文	数 学	英 语	地 理
张三	80	85	94	87
李四	84	87	86	89
王五	87	98	74	89
刘王	78	88	89	84
赵李	85	86	90	87

表 7-2 学生成绩及平均成绩表

姓 名	语 文	数 学	英 语	地 理	平 均 分
张三	80	85	94	87	86.5
李四	84	87	86	89	86.5
王五	87	98	74	89	87
刘王	78	88	89	84	84.75
赵李	85	86	90	87	87

上述表格中显然都用变量去存储这些数据很复杂，从表中可以看出每一列的数据类型都是相同的，因此可以利用数组来存储每一列的数据，再进行计算。在下面的内容中，将会介绍数组的概念及其使用。

7.3 一维数组

一维数组是长度固定的数组，其存储空间是一片连续的区域。本节将讲解一维数组的概念及其应用。

7.3.1 一维数组的声明和初始化

数组在使用之前必须首先进行声明，在本节中将分两部分介绍一维数组：声明和初始化。

1. 一维数组的声明

数组声明方式与变量声明类似，只是数组的声明比变量的声明后面多一个用方括号括起来的数组大小，其声明形式如下：

```
类型 数组名[长度];
```



例如：

```
int x[100];
```

表示声明了一个长度为 100 的整型 x 数组，其中每一个元素的数据类型都为整型。其存储结构如表 7-3 所示。

表 7-3 x 数组存储结构

下标（序号）	相对存储位置	数组元素
1	0	x[0]
2	2	x[1]
3	4	x[2]
:	:	:
100	198	x[99]

x 数组所占字节数=数组元素个数（100）×数据类型字长（2）

2. 一维数组的初始化

一维数组的初始化是指在定义数组时就给数组赋予初始值。一维数组初始化的形式如下：

```
类型 数组名[长度]={数值};
```

其中数值中的每个数据要用逗号分开，例如：

```
int x[10]={0,1,2,3,4,5,6,7,8,9};
```

上述初始化为对整个数组的赋值，也可以单独对每个数组中的元素赋值。例如：

```
x[0]=0;x[1]=1;x[2]=2;...;x[9]=9;
```

在初始化数组的过程中，应注意以下 4 个方面：

（1）若对数组中的所有元素都赋予了初始值，可以不用指定数组的大小，系统将自动根据赋值的个数来确定数组的大小，例如：

```
int x[]={1,2,3,4,5};
```

上述程序中，系统根据其赋值的个数自动将 x 数组大小确定为 5。

（2）若只对数组中的部分元素赋予初始值，则系统会自动为其他元素赋初始值 0。例如：

```
int x[10]={1,2,3,4,5};
```

系统会自动将 x[5]~x[9]赋予初始值 0。

（3）若只声明数组，而不为数组赋值，则数组中的元素值是不确定的。例如：

```
int x[10];
```

x[0]~x[9]中的元素值不能确定。

（4）C 语言数组的大小只能是常量，而不能使用变量，如下面对数组的声明是错误的。

```
int i=100;
```

```
int a[i];
```

7.3.2 一维数组的引用

一维数组声明后，要使用数组中的元素则需引用，其引用形式如下：

```
数组名[下标];
```

下标即要引用的元素在数组中的位置。一个大小为 n 的数组下标可以为 $0\sim n-1$ ，其中 0 为数组的上界， $n-1$ 为数组的下界。例如：

$x[1]$ 和 $x[5]$ 分别为引用 x 数组的第 2 个元素和第 6 个元素。

【范例 7.2】将 10 个数 1, 3, 5, 6, 7, 34, 67, 22, 56, 76 存于数组中，求出这 10 个数的平均数，并将结果输出至屏幕。

分析：本例中给出的 10 个数都为整型数，可用大小为 10 的整型数组存放这 10 个数。定义一个变量计算其总和，另外一个变量用于计算其平均值，最后输出该变量即为这 10 个数的平均值。

范例 7.2 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int x[10]={1,3,5,6,7,34,67,22,56,76}; /*定义一个整型数组 x 并对其进行了初始化*/
05      int i,sum=0;                                  /*定义整型变量 i 和 sum, sum 赋初始值 0*/
06      float aver;
07      for(i=0;i<10;i++)
08          sum=sum+x[i];                             /*计算 x 数组中每个元素的累加和*/
09          aver=sum/10.0;                             /*求 10 个数的平均数*/
10      printf("the average is %f\n",aver);
11  }
```

【代码分析】本题是数组应用的简单范例，详细代码分析如下：

- 第 4 行，定义了一个整型 x 数组，其长度为 10，并对该数组进行了初始化。
- 第 5 行，定义了两个整型变量 i 和 sum ，其中 i 用于 for 循环中作循环变量， sum 用于计算数组中元素的累加和。
- 第 6 行定义一个浮点型变量 $aver$ ，用于保存计算得出的平均数。
- 第 7~8 行为 for 循环，通过 for 循环计算数组中各个元素的和，并将结果保存至 sum 中。
- 第 9 行，将数组中元素的累加和除以 10，即为该数组的平均数。

【运行结果】该程序的执行结果如图 7-1 所示。

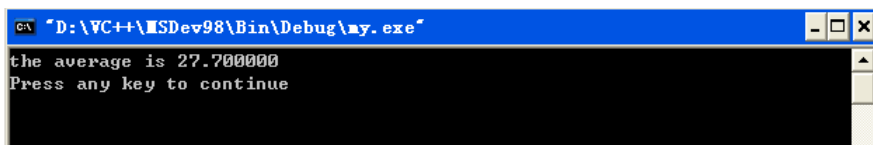


图 7-1 范例 7.2 结果图



【范例 7.3】现有一数组，将数组按逆序输出。

分析：要将数组中的元素实现逆序输出，例如 3，5，2，7，10 逆序输出为 10，7，2，5，3，可定义另外一个长度一样的数组，将原来数组中的元素逆序赋值给该数组，最后输出即可。

范例 7.3 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[5]={3,5,2,7,10},i,y[5];
05      for(i=4;i>=0;i--)                /*for 循环控制 i 从 4 变化至 0*/
06          y[4-i]=x[i];                /*将数组 x 中元素逆序存储在数组 y 中*/
07      for(i=0;i<5;i++)
08          printf("%d ",y[i]);
09      printf("\n");
10  }
```

【代码分析】本例实现数组值的逆序输出，详细代码分析如下：

- 第 5~6 行为 for 循环，将数组 x 中的数据逆序存储至 y 数组中。

【运行结果】该程序的执行结果如图 7-2 所示。

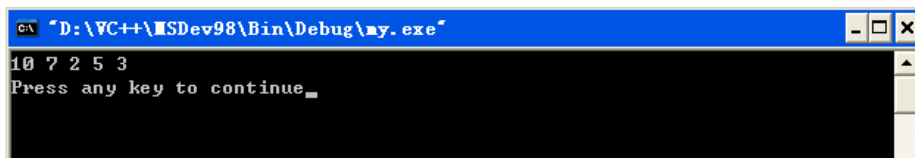


图 7-2 范例 7.3 结果图



注意：上述程序通过另一个数组存储数组的逆序实现数组元素的逆序。其实只要将数组 x 中的元素倒着输出即可。

【范例 7.4】用数组实现输出 Fibonacci 数列的前 20 项。Fibonacci 数列：1，1，2，3，5，8…。

分析：Fibonacci 数列前两项为 1，后一项等于前两项之和，可用数组求解。令 x 数组中的 x[0]和 x[1]为 1， $x[n]=x[n-1]+x[n-2]$ ($x \geq 2$)即可求解第 n 个元素。利用该公式计算 Fibonacci 数列的前 20 个数，进行输出即可。

范例 7.4 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i;
05      int x[20]={1,1};                /*定义数组 x 并初始化部分元素*/
06      for(i=2;i<20;i++)
07          x[i]=x[i-1]+x[i-2];          /*数组 x 后一项等于前两项之和*/
```



```
08     for(i=0;i<20;i++)
09         printf("%d ",x[i]);
10     }
```

【代码分析】本例利用数组求解 Fibonacci 数列，详细代码分析如下：

- 第5行，定义一个长度为20的整型x数组，对数组的前两个数赋值0，其他的元素系统将赋值为0。
- 第6~7行为for循环，将前两个数的和赋给相应的元素，即求得 Fibonacci 数列。

【运行结果】该程序的执行结果如图7-3所示。

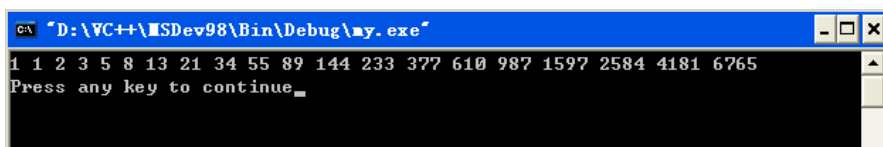


图 7-3 范例 7.4 结果图

【范例 7.5】从键盘输入10个数，将这10个数从小到大排列输出至屏幕。

分析：从键盘输入数据至数组中，可通过scanf()函数实现。对数组中的元素排序可采用冒泡排序法来实现。冒泡排序的思想为：将要排序的数据依次进行相邻元素的比较，若为从小到大排列，则将相邻元素的小者放前面，大者放后面，这样值小的元素就会逐步升至元素的起始位置。对待排序数据进行一次比较，则称为一趟冒泡。第一趟冒泡可将最小值放置在待排序数据的起始位置，第二趟排序可将剩余数据中最小的元素放于第二个位置。n个待排序的元素经过n-1排序即可得到正确的排序序列。

范例 7.5 代码实现

```
01     #include <stdio.h>
02     void main()
03     {
04         int x[10],i,j,k;
05         printf("input the number\n");
06         for(i=0;i<10;i++)
07             scanf("%d",&x[i]);          /*调用 scanf() 函数从键盘获取字符保存至数组中*/
08         for(i=0;i<9;i++)                /*嵌套 for 循环*/
09             for(j=9;j>i;j--)
10                 if(x[i]>x[j])            /*若 x[i] 大于 x[j], 则交换 x[i] 和 x[j] 的值*/
11                 {
12                     k=x[i];
13                     x[i]=x[j];
14                     x[j]=k;
15                 }
16         printf("the sort number is: ");
17         for(i=0;i<10;i++)                /*利用 for 循环和 printf() 函数输出已排好序的数组*/
18             printf("%d ",x[i]);
19     }
```




【代码分析】本例利用数组对数据采用冒泡法进行排列，详细代码分析如下：

- 第 6、7 行，通过 for 循环和 scanf() 函数实现 10 个数据的输入。
- 第 8~15 行为冒泡排序，总共进行 9 趟比较，第 1 趟比较将最小的数放置于数列的最前方即 x[0] 中，第 2 趟比较将剩余数中的最小者放于 x[1]。依此类推，经过 9 趟比较后数列即可排好序。

【运行结果】该程序的执行结果如图 7-4 所示。

```
C:\ D:\Win-TC\projects\noname.exe
input the number
-23
34
55
67
-12
32
56
78
-54
87
the sort number is:-54 -23 -12 32 34 55 56 67 78 87 _
```

图 7-4 范例 7.5 结果图



提示：

本题中数列的初始状态如下：

```
-23 34 55 67 -12 32 56 78 -54 87
第一趟排序后：
-54 34 55 67 -12 32 56 78 -23 87
第二趟排序后：
-54 -23 55 67 -12 32 56 78 34 87
第三趟排序后：
-54 -23 -12 67 55 32 56 78 34 87
第四趟排序后：
-54 -23 -12 32 55 67 56 78 34 87
第五趟排序后：
-54 -23 -12 32 34 67 56 78 55 87
第六趟排序后：
-54 -23 -12 32 34 55 56 78 67 87
第七趟排序后：
-54 -23 -12 32 34 55 56 78 67 87
第八趟排序后：
-54 -23 -12 32 34 55 56 67 78 87
第九趟排序后：
-54 -23 -12 32 34 55 56 67 78 87
```



7.4 二维数组

在实际应用中，有很多问题是二维或多维的，用一维数组很难解决。例如，要计算一个部门 30 个员工的工资，可以用一个一维数组计算，若要计算全公司 20 个部门的员工工资，假设

每个部门有 30 个员工，则需用 20 个不同的一维数组，用一维数组来实现这个问题会相当麻烦。此时可采用多维数组解决上述问题：计算公司 20 个部门的员工工资，将定义为一个二维数组（部门，人数）。

7.4.1 二维数组的声明和初始化

二维数组有两个下标，与一维数组一样，数组中的元素必须为同一类型。二维数组的声明形式如下所示。

```
类型 数组名[常量][常量];
```

例如：

```
int x[5][4];
```

上述程序中声明了一个具有 20 个连续存储单元的 x 数组，其中第一维的长度为 5，第二维的长度为 4，每个元素的类型都为整型。其描述形式如表 7-4 所示。

表 7-4 二维数组x形式表

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]
x[3][0]	x[3][1]	x[3][2]	x[3][3]
x[4][0]	x[4][1]	x[4][2]	x[4][3]

在二维数组中，通常把第一个下标称做行下标，第二个下标称做列下标。二维数组中的元素在计算机中是连续存储的，从第[0][0]个元素开始。其存储结构如表 7-5 所示。

表 7-5 二维数组x存储结构表

序 列 号	相对存储位置	x 数组中的元素
1	0	x[0][0]
2	2	x[0][1]
3	4	x[0][2]
4	6	x[0][3]
5	8	x[1][0]
6	10	x[1][1]
7	12	x[1][2]
8	14	x[1][3]
9	16	x[2][0]
10	18	x[2][1]
11	20	x[2][2]
12	22	x[2][3]
13	24	x[3][0]
14	26	x[3][1]



续表

序 列 号	相对存储位置	x 数组中的元素
15	28	x[3][2]
16	30	x[3][3]
17	32	x[4][0]
18	34	x[4][1]
19	36	x[4][2]
20	38	x[4][3]

二维数组存储所占字节数=行数×列数×数据类型所占字节数。在使用二维数组的过程中，应注意以下 5 个方面：

- (1) 数组元素在内存中的存放是连续的，不是分开的。
- (2) 若有一个大小为 $m \times n$ 的二维数组 a ，则 $a[x][y]$ 在数组中的位置为 $x \times n + y$ （二维数组从 $a[0][0]$ 开始存储）。
- (3) 要引用二维数组中的一个元素，则需标明其行下标和列下标。例如：
 $a[3][2]$ 为引用第 4 行第 3 列的元素。
- (4) 二维数组可以看成是由一维数组组成的数组。例如：
 $\text{int } a[3][2]$ 可以看成 3 个长度为 2 的一维数组组成的数组。
- (5) 二维数组的初始化。二维数组进行初始化时需对每行分别进行赋值，并且用花括号括起来。例如：

```
int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

7.4.2 二维数组应用举例

【范例 7.6】从键盘输入一个 4×4 矩阵，求其主对角线元素之和。

分析：利用 for 循环求主对角线的元素之和，保存至变量中，最后输出结果即可。

范例 7.6 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[4][4];                /*定义二维数组 x*/
05      int i,s=0;
06      for(i=0;i<4;i++)
07          scanf("%d%d%d%d",&x[i][0],&x[i][1],&x[i][2],&x[i][3]);
08      printf("the array\n");
09      for(i=0;i<4;i++)
10          printf("%d %d %d %d\n",x[i][0],x[i][1],x[i][2],x[i][3]);
11                                     /*输出二维数组 x 中的各个元素*/
12      for(i=0;i<4;i++)
13          s=s+x[i][i];                /*通过 for 循环计算数组各个元素的累加和*/
14      printf("the sum is %d",s);
15  }
```

- 【代码分析】本例为二维数组的简单范例，详细代码分析如下：
- 第 4 行，定义了一个 4×4 大小的二维矩阵 x，用来保存从键盘输入的数。
 - 第 5 行，定义了两个整型变量，其中变量 i 用于作为 for 循环变量，变量 s 用于保存计算得出的总和。
 - 第 9、10 行，用 for 循环和 printf()函数输出该数组至屏幕。
 - 第 11、12 行，用 for 循环计算主对角线元素的累加和，并保存至变量 s 中。
- 【运行结果】该程序的执行结果如图 7-5 所示。

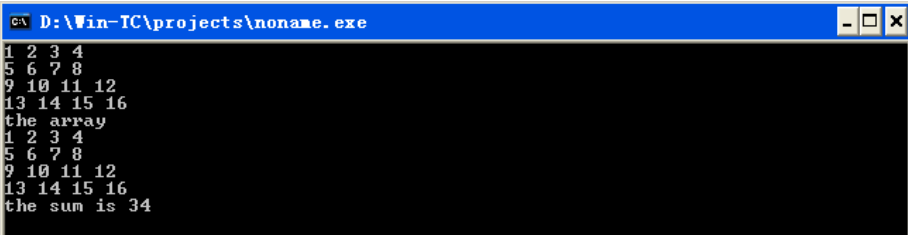


图 7-5 范例 7.6 结果图

【范例 7.7】从键盘输入一个 4×3 的矩阵，输出数组中最大的元素及其所在的行和列。

分析：利用 scanf()函数从键盘输入矩阵，保存至二维数组中。因为矩阵大小为 4×3，因此二维数组大小可定义为 4×3。利用 for 循环逐个对二维数组中的元素进行比较，找出其中最大的元素，再输出该元素及其行和列。

范例 7.7 代码实现

```

01  #include <stdio.h>
02  void main()
03  {
04      int x[4][3],i,j,max=0,y,z;    /*定义整型二维数组 x 及变量 i,j,max,y,z*/
05      for(i=0;i<4;i++)
06          scanf("%d%d%d",&x[i][0],&x[i][1],&x[i][2]);
07      printf("the array\n");
08      for(i=0;i<4;i++)
09          printf("%d %d %d\n",x[i][0],x[i][1],x[i][2]);
10      for(i=0;i<4;i++)                /*嵌套 for 循环结构*/
11          for(j=0;j<3;j++)
12              if(max<x[i][j])          /*若 max 小于 x[i][j],则将 x[i][j]赋值给变量 max*/
13                  { max=x[i][j];y=i,z=j;}
14      printf("the max is %d\n",max);    /*输出数组中的最大元素*/
15      printf("position:%d,%d",y+1,z+1);
16  }
  
```

- 【代码分析】本题利用二维数组求最大元素，详细代码分析如下：
- 第 4 行，定义了一个二维数组和 5 个整型变量，其中二维数组 x 用来保存从键盘键入的矩阵，i 和 j 作为 for 循环的控制变量，max 用来保存数组中最大的元素，y 和 z 用来记录最大元素的行下标和列下标。



由浅入深学 C 语言——基础、进阶与必做 430 题

- 第 5、6 行，用 for 循环和 scanf() 函数实现矩阵的输入，其中每次输 3 个数，中间用空格分开。
- 第 10~13 行，利用嵌套 for 循环寻找数组中的最大元素。令数组中的元素依次与 max 的值进行比较，若比 max 中的值大则赋值给 max，同时记录其行下标和列下标。依次计算，将数组中的元素都比较一遍即可得到数组中的最大元素。

【运行结果】该程序的执行结果如图 7-6 所示。

```
D:\Win-TC\projects\noname.exe
23 34 2
45 54 5
78 65 42
68 23 57
the array
23 34 2
45 54 5
78 65 42
68 23 57
the max is 78
position:3,1
```

图 7-6 范例 7.7 结果图



注意：第 15 行中输出最大元素的位置用 y+1 和 z+1 来计算是因为数组的行下标和列下标都是从 0 开始的，而实际中人们计算是从 1 开始的，因此要加 1。

【范例 7.8】编写一个程序，从键盘输入两个矩阵，计算两个矩阵的乘积并输出。

分析：设有两个大小分别为 $a \times b$ 和 $c \times d$ 的矩阵，则两个矩阵要相乘，必须满足 $b=c$ ，同时得到大小为 $a \times d$ 的矩阵。若要相乘的条件，则利用公式 $Z_{ad}=X_{ab} \times Y_{bd}$ 计算矩阵相乘的结果，最后再利用输出函数输出该结果至屏幕。

范例 7.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[2][3],y[3][2],z[2][2];          /*定义整型二维数组 x,y,z*/
05      int i,j,k,s;
06      for(i=0;i<2;i++)
07          scanf("%d%d%d",&x[i][0],&x[i][1],&x[i][2]);
08      for(j=0;j<3;j++)
09          scanf("%d%d",&y[j][0],&y[j][1]);
10      for(i=0;i<2;i++)
11          for(j=0;j<2;j++)
12          {
13              s=0;
14              for(k=0;k<3;k++)              /*计算矩阵 x 和矩阵 y 的乘积*/
15                  s=s+x[i][k]*y[k][j];
16                  z[i][j]=s;                /*将结果保存至数组 z 中*/
17          }
18      for(i=0;i<2;i++)
```

```

19     for(j=0;j<2;j++)
20         printf("z[%d][%d]=%3d\n",i,j,z[i][j]);
21     }

```

【代码分析】本例利用数组求矩阵的乘积，详细代码分析如下：

- 第 4 行定义了 3 个数组，其中 x，y 数组用来保存从键盘输入的矩阵，z 数组用来保存计算得出的矩阵。
- 第 10~17 行，利用给出的公式计算两个矩阵相乘的乘积。

【运行结果】该程序的执行结果如图 7-7 所示。



图 7-7 范例 7.8 结果图



注意：第 13 行 s=0 这一语句很重要，两个矩阵相乘所得的矩阵为原来两个矩阵及行和列的乘积和，若 s 不赋 0，则下一次计算结果会出错。



7.5 字符数组

在计算机中经常会处理字符，C 语言对字符的处理主要是通过字符数组实现的。在本节中将会讲述字符数组的概念及其应用。

7.5.1 字符串与字符数组

本节分两部分介绍字符数组：声明和初始化。

1. 字符数组的声明

在 C 语言中，由若干字符组成的序列称为字符串。字符串不能用一个变量存放，必须用字符数组来存放。字符串一般以 \0 作为结束标志，\0 表示空的意思，不包含任何字符，只用来作为字符串的结尾标志，例如字符串 hello 存储形式如下：

h	e	l	l	o	\0
---	---	---	---	---	----

字符数组分为一维字符数组和多维字符数组，在 C 语言中运用最多的一般是一维字符数组和二维字符数组。

一维字符数组声明方式与一维整型数组类似，只是类型标识符变为了 char。例如：

```
char c[20];
```



由浅入深学 C 语言——基础、进阶与必做 430 题

定义了一个长度为 20 的字符数组，可以存放 20 个字符。赋值语句如下：

```
c[0]='s';c[7]='w';
```

上述程序中，对 c 数组中的第 1 个元素和第 8 个元素赋值，其中一定要用单引号把字符给包围起来，否则会出错。

二维字符数组的声明也与二维整型数组类似，其声明方式如下：

```
char c[10][10];
```

定义了一个长度为 10×10 的二维字符数组。

2. 字符数组的初始化

一维字符数组的初始化有以下两种方式：

(1) 逐个为字符数组赋值。例如：

```
char c[6]={ 'H', 'e', 'l', 'l', 'o'};
```

系统会自动在字符串的末尾加上\0 标识符用以表示字符串的结束。在计算字符串长度时，\0 不算在内。通常字符数组的大小都要比存储字符的个数多 1，用来存放\0 字符。

(2) 用字符串直接给字符数组赋值。例如：

```
char c[6]={ "Hello"};
```

直接将字符串赋给字符数组来初始化。上述程序也可改为：

```
char c[6]= "Hello";  
char c[]={ "Hello"};
```

这三句赋值语句都是一样的。

二维数组的初始化方式如下，例如：

```
char c[2][6]={ "Hello","World"};
```

定义了一个二维数组 c，存储了两个字符串，分别为“Hello”和“World”，其中每个字符串的长度不得超过 5。

7.5.2 字符串输入、输出函数

在前面的章节中，已经讲解了一些字符的输入、输出函数。例如 scanf()、printf()、getchar()、putchar()等函数，本节将重点讲解字符串输入、输出函数。

1. 逐个字符的输入和输出

(1) 字符输入函数

逐个字符的输入函数有 scanf()和 getchar()两个函数。其示例如下：

```
for(i=0;i<5;i++)  
scanf("%c",&s[i]);
```



```
for(i=0;i<5;i++)
s[i]=getchar();
```

scanf()函数和 getchar()函数一般可以通用。

(2) 字符输出函数

逐个字符的输出函数包括 printf()和 putchar()两个函数。其示例如下：

```
for(i=0;i<5;i++)
printf("%c",s[i]);
for(i=0;i<5;i++)
putchar(s[i]);
```

printf()函数和 putchar()函数一般也可以通用。

2. 字符串的输入和输出

(1) 字符串输入函数

整个字符串的输入函数有 scanf()和 gets()两个函数。其示例如下：

```
scanf("%s",s);
gets(s);
```

上述程序中的 scanf()和 gets()函数可以通用，都能实现整个字符串的输入。

(2) 字符串输出函数

整个字符串的输出函数有 printf()和 puts()函数。其示例如下：

```
printf("%s",s);
puts(s);
```

printf()函数和 puts()函数也可以通用，其功能为输出整个字符串。

【范例 7.9】利用 gets()和 puts()函数实现字符串的输入和输出。

范例 7.9 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      char s[20];                /*定义一个字符数组 s*/
05      gets(s);                  /*调用 gets() 函数从键盘获取字符串*/
06      puts(s);                  /*调用 puts() 函数输出字符串*/
07  }
```

【代码分析】本例为字符串输入、输出函数的应用，详细代码分析如下：

- 第 4 行，定义一个字符数组 s。
- 第 5 行，利用 gets()函数实现字符串的输入。
- 第 6 行，通过 puts()函数输出从键盘输入的字符串。

【运行结果】该程序的执行结果如图 7-8 所示。

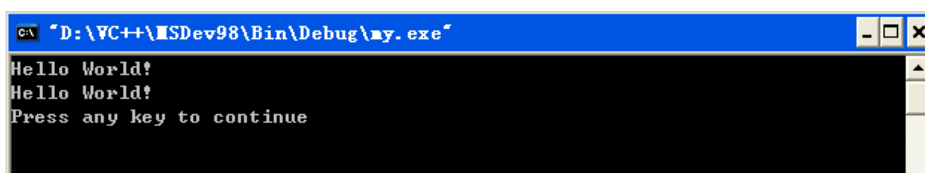


图 7-8 范例 7.9 结果图

7.5.3 字符串函数

C 语言提供了专门的字符串处理函数，包括合并、比较等标准函数。它们都包含在 `string.h` 头文件中，在使用之前必须先包含该头文件。

1. 比较函数 `strcmp()`

`strcmp()` 函数用来比较两个字符串的大小，其调用形式如下：

```
strcmp(s1,s2);
```

该函数对 `s1`、`s2` 两个字符串中的每个字符逐一进行比较，即比较其 ASCII 码值的大小，直到遇到不同的字符或字符串的末尾。若 `s1` 中有一个字符比 `s2` 中的字符大，则认为 `s1` 字符串比 `s2` 字符串大。`s1` 中每个字符都与 `s2` 中字符相等，则称 `s1` 和 `s2` 相等。此函数当 `s1` 小于 `s2` 返回值为负值，当 `s1` 等于 `s2` 返回值为 0，`s1` 大于 `s2` 返回值为正值。

2. 复制函数 `strcpy()`

`strcpy()` 函数用来复制一个字符串，其调用形式如下：

```
strcpy(s1,s2);
```

此函数的功能是将字符串 `s2` 的内容复制给字符串 `s1`。因此 `s1` 的长度必须要超过 `s2` 的长度，才能完整的复制。

C 语言中不能直接将字符串赋给字符数组，如 `s1=s2` 是错误的。要将字符串常量赋给字符数组，则必须使用 `strcpy()` 函数。

3. 合并函数 `strcat()`

`strcat()` 函数用来合并两个字符串，其调用形式如下：

```
strcat(s1,s2);
```

此函数的功能是将字符串 `s2` 连接到字符串 `s1` 的后面。在该函数调用时，会自动去掉 `s1` 中的 `\0` 结束标识符，再将 `s2` 中的内容连接到 `s1` 的后面，最后在 `s1` 的末尾加上 `\0` 标识符用来表示字符串的结束。

例如：

```
#include <string.h>
void main()
{
    char s1[20]= "Hello";
```



```
char s2[6]= "World";
strcat(s1,s2);
puts(s1);
}
```

上述程序中将 s2 连接到 s1 后面，最后 s1 中的字符串变为“HelloWorld”。

4. 字符串长度函数 strlen()

strlen()函数用来求一个字符串的长度，其调用形式如下：

```
strlen(s);
```

此函数的功能是计算字符串的长度，其中不包括字符串的结束标志\0。例如：

```
char s[]="Happy day";
printf("%d",strlen(s));
```

上述程序为求字符串 s 的长度，其输出结果为 9。



7.6 数组实战项目

【范例 7.10】 现有 5 个学生，4 门科目，已知所有学生的各科成绩，计算每个学生的平均成绩和每门科目的平均成绩。

分析：有 5 名学生，4 门科目，可定义为二维数组，又由于科目成绩不一定为整数，因此需定义为浮点型。定义两个数组 x，y 分别计算每个学生的总成绩和每门科目的总成绩，x 数组中每个元素除以 4 即可求出每个学生的平均成绩，y 中每个元素除以 5 即为每门科目的平均成绩。

范例 7.10 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
    /*定义二维数组 score 并进行了初始化*/
04  float score[5][4]={77,83,84,79},{70,80,84,90},{76,91,88,84},{60,76,69,71},
    {76,77,80,70}};
05  float x[5]={0,0,0,0,0},y[4]={0,0,0,0};
06  int i,j;
07  for(i=0;i<5;i++)          /*通过嵌套 for 循环计算每门学生总成绩*/
08  for(j=0;j<4;j++)
09      x[i]=x[i]+score[i][j];
10  for(j=0;j<4;j++)
11  for(i=0;i<5;i++)
12      y[j]=y[j]+score[i][j];
13  for(i=0;i<5;i++)          /*输出每门学生的平均成绩*/
```



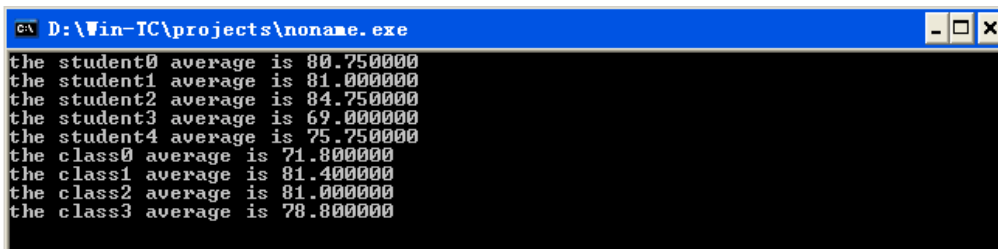
由浅入深学 C 语言——基础、进阶与必做 430 题

```
14  printf("the student%d average is %f\n",i,x[i]/4);
15  for(j=0;j<4;j++)          /*输出每门科目的平均成绩*/
16  printf("the class%d average is %f\n",j,y[i]/5);
17  }
```

【代码分析】本例为二维数组的应用范例，详细代码分析如下：

- 第 4 行，定义了一个浮点型数组，并将每个学生的各科成绩赋值给数组。
- 第 5 行，定义了两个浮点型数组，用于统计每个学生的总成绩和每门科目的总成绩，从而计算其平均成绩。
- 第 7~9 行，统计每个学生的总成绩。
- 第 10~12 行，统计每门科目的总成绩。

【运行结果】该程序的执行结果如图 7-9 所示。



```
C:\ D:\Win-TC\projects\noname.exe
the student0 average is 80.750000
the student1 average is 81.000000
the student2 average is 84.750000
the student3 average is 69.000000
the student4 average is 75.750000
the class0 average is 71.800000
the class1 average is 81.400000
the class2 average is 81.000000
the class3 average is 78.800000
```

图 7-9 范例 7.10 结果图

【范例 7.11】从键盘输入一个字符串，删除该字符串中指定位置上的字符并输出。

分析：删除字符串指定位置上的字符，可以用 `strcpy(&s[n],&s[n+1])` 使第 n 个字符后面的字符都向前移动一个位置，即可覆盖第 n 个字符。

范例 7.11 代码实现

```
01  #include <stdio.h>
02  #include <string.h>          /*包含 string.h 头文件*/
03  void main()
04  {
05      char s[50];
06      int i,j,n;
07      printf("input the string: ");
08      gets(s);
09      printf("input the delete char: ");
10      scanf("%d",&n);          /*输入要删除的字符的位置*/
11      strcpy(&s[n-1],&s[n]);    /*通过 strcpy() 函数覆盖掉第 n 个字符*/
12      printf("the change string is: ");
13      puts(s);                /*输出删除字符后的字符串*/
14  }
```

【代码分析】本题利用 `strcpy()` 函数删除字符，详细代码分析如下：

- 第 8 行，利用 `gets()` 函数实现字符串 `s` 的输入。

- 第 10 行，输入要删除的字符所在的位置，并将其保存至变量 n 中。
- 第 11 行，用 `strcpy()` 函数实现字符的删除操作。它将第 n 个字符后面的字符串都往前移一位，覆盖第 n 个字符即实现删除第 n 个字符。

【运行结果】该程序的执行结果如图 7-10 所示。

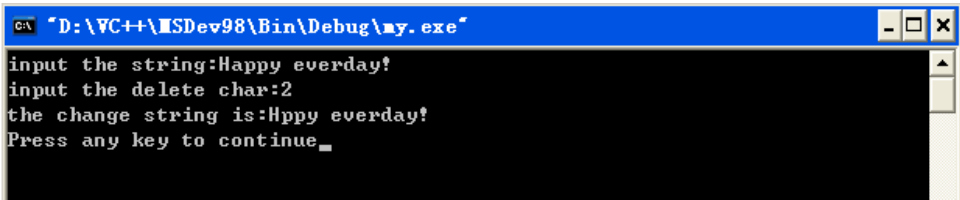


图 7-10 范例 7.11 结果图

【范例 7.12】从键盘输入 10 个整数，将其从小到大排列再输出至屏幕。

分析：对 10 个数进行排列可以采用前面讲过的冒泡排序法，也可以采用另一种方法：选择排序法。选择排序法的思想是：首先设置一个变量保存元素下标，将该元素与其他元素进行比较。若大于其他元素则交换其下标，比较完后再判断其下标是否发生变化，若变化则交换两个元素的值，否则不变。经过一次比较，即可得出数列中的最小元素并将其放在数列的首位。依此类推，经过 $n-1$ 次比较即可得到 n 个元素从小到大的序列。

范例 7.12 代码实现

```

01  #include <stdio.h>
02  void main()
03  {
04      int x[8];
05      int i,j,k,temp;
06      for(i=0;i<8;i++)                /*利用 for 循环将数据保存至数组 x 中*/
07          scanf("%d",&x[i]);
08      for(i=0;i<7;i++)                /*选择排序过程*/
09      {
10          k=i;
11          for(j=i+1;j<8;j++)
12              if(x[i]>x[j])            /*若 x[i]大于 x[j]*/
13                  k=j;                /*记录 j 下标*/
14          if(k!=i)                    /*若 k 与 i 值不同*/
15              {temp=x[i];x[i]=x[k];x[k]=temp;} /*交换 x[i]和 x[k]的值*/
16      }
17      for(i=0;i<8;i++)
18          printf("%4d",x[i]);
19  }
```

【代码分析】本例利用选择排序法对数组进行排序，详细代码分析如下：

- 第 5 行，定义 4 个整型变量。其中 i 和 j 用于控制 `for` 循环的执行次数， k 记录数组元素的下标，`temp` 变量用于元素值的交换。
- 第 8~16 行，用选择排序法对数组进行排序。先用 k 记录每个元素的下标，在将该元



素与后面的元素进行比较，若大于后面的元素则改变 k 的值，即记录其下标。最后判断 k 的值是否与 i 的值相等，若不相等则交换这两个元素的值即将该数列中的最小元素放置在最前面的位置，因此总共进行 7 次比较即可得到 8 个元素从小到大的排列。

【运行结果】该程序的执行结果如图 7-11 所示。

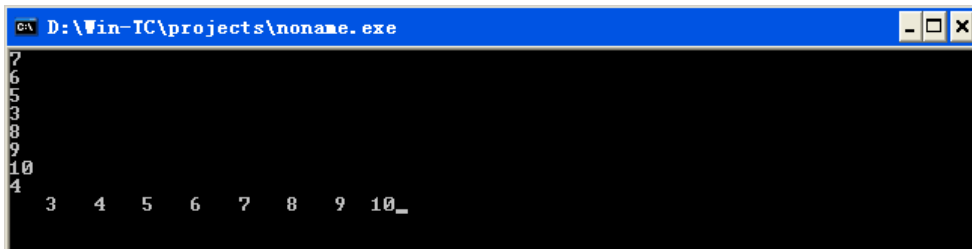


图 7-11 范例 7.12 结果图



7.7 小结

在本章中讲解数组的概念及其应用，其中包括整型数组、浮点型数组和字符数组等。在 C 语言程序的学习过程中经常会使用到数组，因此读者应认真掌握数组的概念及其应用。



7.8 习题

一、选择题

1. 以下程序运行后，其结果为（ ）。

```
void main()
{
    char s[5]= "everyday" ,i;
    for(i=0;i<5;i++)
        printf("%c ",s[i]);
    getch();
}
```

- A. every
- B. ever
- C. everyday
- D. 程序有误，不能运行

【提示】字符数组定义时其大小为 5，后又对其赋值字符串 `everyday`，字符串的长度很明显大于 5，因此程序运行会出错。

2. 若有以下程序，输入 `cd`，其运行结果为（ ）。

```
void main()
{
    char s[2];
```



```
gets(s);
printf("%c,%c",s[1],s[2]);
}
```

A. c,d

B. d,

C. c,

D. cd,

【提示】本题中定义了一个字符数组 `s`，其长度为 2，因此该数组只能存放两个字符，并且其元素的下标在 0~1 之内。上述程序中，`s[2]` 已经超出了数组 `s` 的范围，其存放的为字符串结束标志 `\0`，因此本题的输出结果为 `d`。

3. 下列语句不正确的是 ()。

A. `int a[]={1,2,3};`B. `int a[5]={1,2,3};`C. `int a[];`D. `int a[5];`

【提示】A 选项中虽没有指明数组的大小，但对其进行了初始化。系统会自动根据赋值的个数确定其大小，因此 A 选项正确。B 选项中，定义了一个长度为 5 的整型数组，对前三个元素赋了值，后面两个元素系统自动赋值为 0，正确。C 选项中没有指明数组的大小，因此错误。D 选项定义了长度为 5 的整型数组 `a`，正确。

4. 下列程序的运行结果是 ()。

```
void main()
{
    char s1[6],s2[]="first";
    strcpy(s1,s2);
    printf("%s",s1);
}
```

A. first

B. f

C. 运行有误

D. 都不正确

【提示】本题中定义了字符数组 `s2`，并将字符串 `first` 赋给 `s2`。利用 `strcpy()` 函数复制给字符数组 `s1`，因此 `s1` 的值最终为 `first`。

5. 以下程序运行结果为 ()。

```
void main()
{
    int x[5],i;
    for(i=0;i<5;i++)
        x[i]=2*i+1;
    for(i=0;i<5;i++)
        printf("%d",x[i]);
}
```

A. 13579

B. 02468

C. 1357

D. 0246

【提示】本题利用 `for` 循环将 `2*i+1` 的值赋给数组 `x`，输出数组 `x` 中的每个元素。



6. 以下程序运行结果为 ()。

```
void main()
{
    char str[]="abcdrf",s;
    int i;
    for(i=2;(s=str[i])!=0;i++)
    {
        switch(s)
        {
            case 'd':++i;break;
            case 'l':continue;
            default:putchar(s);continue;
        }
    }
}
```

A. cd

B. cf

C. cdef

D. ef

【提示】本题将字符数组中各个元素与 switch 结构中特定值比较，若为字符 d 则执行++i 操作，若为字符 l 则返回 for 循环继续执行，否则输出该字符。

7. 以下程序运行结果为 ()。

```
void main()
{
    int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    int i,s=0;
    for(i=0;i<3;i++)
        s=s+a[i][2];
    printf("%d",s);
}
```

A. 20

B. 21

C. 22

D. 19

【提示】本题将数组 a[3][4]中的每一行第 3 个元素相加，最后输出结果。

8. 以下程序运行结果为 ()。

```
void main()
{
    int x=20;
    int i=0,s[5];
    do
    {
        s[i]=x%2;
        i++;
        x=x/2;
    }while(i<5);
}
```



```
for(i=0;i<5;i++)
printf("%d",s[i]);
}
```

A. 00001

B. 00100

C. 00101

D. 11001

【提示】本题 do-while 循环中将 $x\%2$ 的余数保存至数组 s 的各个元素中，然后将 x 整除 2 的值赋给变量 x ，最后输出数组 s 中的元素。

9. 以下程序运行结果为 ()。

```
void main()
{
    int i,j=3;
    int a[5];
    for(i=0;i<5;i++)
    {
        a[i]=j*2+2;
        j++;
    }
    for(i=0;i<5;i++)
    printf("%d",a[i]);
}
```

A. 810121416

B. 791113

C. 123456

D. 7891011

【提示】本题 for 循环控制变量 j 从 0 变化至 5，将 $j*2+2$ 的值赋给数组 a 中的各个元素。

二、填空题

1. 以下程序的运行结果为_____。

```
#include <stdio.h>
#include <string.h>
void main()
{
    char x[10]= "abc",y[5]= "def";
    strcat(x,y);
    puts(x);
}
```

【提示】strcat()函数的功能为将字符串拼接到另一个字符串上。

2. 以下程序的运行结果为_____。

```
void main()
{
    int i,j,x[10]={4,5,6,7,1,2,3,8,9,10};
    for(i=0;i<9;i++)
    for(j=i;j<10;j++)
```




```
if(x[i]<x[j])
    x[i]=x[j];
for(i=0;i<10;i++)
    printf("%d", x[i]);
}
```

【提示】本题利用冒泡排序法对数组排序，但交换过程中没有使用临时变量，因此值大的元素将会覆盖值小的元素。

3. 以下程序的功能为找出 10 个数中最小的数及其下标，并将结果输出，请补充下面程序的空白处。

```
void main()
{
    int a[10],i,min=-32768,k;
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(i=0;i<10;i++)
        if( )
        {
            min=a[i];
        }
    printf("the min is %d,a[%d] ",_____,k);
}
```

【提示】本题中利用 for 循环将变量 min 值与数组中每个元素进行比较，若 min 值大于数组中的元素，则将该元素赋值给 min。

4. 有以下程序，其功能为判断输入的十个字符串的长度，并输出最长的字符串。补充下面程序的空白处。

```
#include <stdio.h>
#include <string.h>
void main()
{
    char s[10][80],c[80];
    int i;
    for(i=0;i<10;i++)

    for(i=0;i<10;i++)
        if(strlen(c)<strlen(s[i]))
            ;
    printf("%s",c);
}
```

【提示】本题将第一个字符串长度赋值给变量 c，然后将该长度与剩余字符串的长度比较。若该值小于其他字符串的长度，则赋值为该字符串。

5. 以下程序运行结果为_____。



```
void main()
{
    int i,j;
    int a[3][4]={1,2,3,4,5,6,7,8,9};
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
}
```

【提示】本题利用嵌套 for 循环分行输出数组中的每一个元素。

6. 以下程序运行结果为_____。

```
void main()
{
    int a[]={1,2,3,4,5,6},*p=a;
    p++;
    printf("%d",*(p+4));
}
```

【提示】本题中指针 p 指向数组 a 首地址，p++后指针 p 指向数组第 2 个元素，因此 p+4 指向数组第 6 个元素。

7. 以下程序运行结果为_____。

```
void main()
{
    char s[]="abcdefgh";
    char *p;
    p=s;
    while(*p!= '\0')
    {
        printf("%c",*p);
        p=p+2;
    }
}
```

【提示】本题利用指针 p 指向数组 s 的首地址，输出*p 中的内容，然后 p=p+2 继续输出直到指针 p 移动到字符串的末尾为止。

8. 以下程序运行结果为_____。

```
void main()
{
    int i,j,a[5];
    for(i=0;i<5;i++)
```



```
{
    j=i;
    a[i]=3*j+1;
}
for(i=0;i<5;i++)
printf("%d",a[i]);
}
```

【提示】本题利用 for 循环将 $3j+1$ 赋值给数组 a，然后进行输出。

9. 以下程序的运行结果为_____。

```
void main()
{
    int i,j,a[2][4]={1,2,3,4,5,6,7,8};
    for(i=0;i<2;i++)
    {
        j=i;
        a[i][j]=0;
    }
    for(i=0;i<2;i++)
    for(j=0;j<4;j++)
    printf("%d",a[i][j]);
}
```

【提示】本题中通过 for 循环将主对角线上的元素都赋值为 0，然后进行输出。

10. 以下程序的运行结果为_____。

```
void main()
{
    int i,j;
    int a[5]={1,2,3,4,5};
    for(i=0;i<5;i++)
    for(j=0;j<i;j++)
    a[i]=a[j]-a[i];
    for(i=0;i<5;i++)
    printf("%d",a[i]);
}
```

【提示】本题利用二重 for 循环将 $a[j]-a[i]$ 的值赋给变量 a[i] 再进行输出。

11. 以下程序运行结果为_____。

```
void main()
{
    char s1[]="Hello world! ",s2[50];
    char *p=s1;
    int i=0;
    while(*p!= '\0')
    {
```



```
s2[i]=*p;
p++;
i++;
}
printf("%s",s2);
}
```

【提示】上述程序定义了一个字符数组 s1 并进行了初始化，利用指针 p 将字符数组 s1 中的内容复制到字符数组 s2 中。

12. 以下程序的运行结果为_____。

```
void main()
{
    int a[10],i,s=0;
    for(i=0;i<10;i++)
        a[i]=2*i-1;
    for(i=0;i<10;i++)
        s=s+a[i];
    printf("%d",s);
}
```

【提示】利用 for 循环将 $2i-1$ 赋值给数组 a，然后计算其累加和，最后输出结果。

13. 以下程序运行结果为_____。

```
void main()
{
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    int i;
    for(i=0;i<10;i++)
    {
        printf("第%d个元素为%d",i+1,a[i]);
        printf("\n");
    }
}
```

【提示】上述程序中定义了一个数组 a 并进行了初始化，然后输出数组中每个元素。

三、编程题

1. 设有以下 10 个数 (4, 2, 7, 8, 9, 10, 11, 24, 5, 6)，编写程序用选择排序法使这 10 个数按升序排序，并输出排序后的结果。

【提示】前面已经讲过选择排序法的概念。选择排序法：设置一个变量保存元素下标，将该元素与其他元素进行比较。若大于其他元素则交换其下标，比较完后再判断其下标是否发生变化，若变化则交换两个元素的值，否则不变。经过一次比较，即可得出数列中的最小元素并将其放在数列的首位。依此类推，经过 $n-1$ 次比较即可得到 n 个元素从小到大的序列。

【核心代码】



```
. ...
for(i=0;i<9;i++)
{
    k=i;
    for(j=i+1;j<10;j++)
        if(x[i]>x[j])
            k=j;
    if(k!=i)
        {temp=x[i];x[i]=x[j];x[j]=temp;}
}
```

2. 从键盘输入 30 个整数，统计并输出这些数字中正数的个数。

【提示】从键盘输入 30 个整数，可以用整型数组来保存这些整数。设置一个计数器，初始值为 0，利用循环对数组内的元素依次判断，看其值是否大于 0，若大于 0 则为正数计数器加 1，否则计数器的值不变。数组内的元素都判断过后，计数器的值即为正数的个数。

【核心代码】

```
for(i=0;i<30;i++)
scanf("%d",&a[i]);
for(i=0;i<30;i++)
if(a[i]>0)
count++;
printf("the num is %d",count);
```

3. 从键盘输入一个字符串，编写一个程序实现从键盘输入一个数，可删除字符串该位置上的字符并输出改变后的字符串。

【提示】从键盘输入字符串，可用字符数组来保存。要删除字符串固定位置上的字符，可以使该字符后面的字符都往前移一位即可，即覆盖掉该字符。本题可以利用 `strcpy()` 函数实现，也可以不利用该函数实现。

【核心代码】

```
gets(s);
scanf("%d",&n);
k=strlen(s);
for(i=n;i<k;i++)
s[i-1]=s[i];
s[k-1]='\0';
puts(s);
```

4. 编写一个程序，实现从键盘输入 10 个整数，将其逆序输出。

【提示】利用 `scanf()` 函数获取从键盘输入的 10 个整数，保存至数组中。有两种方法可以实现逆序输出，一种是将整数逆序保存至数组中，另一种为将整型保存至数组中再逆序输出。

【核心代码】

```
for(i=0;i<10;i++)
scanf("%d",&a[i]);
```



```
for(i=10;i>=0;i--)  
printf("%d ",a[i]);
```

5. 现有一个长度为 4×5 的二维数组, 要求将该数组的行和列元素互换, 保存至另外一个数组中, 并输出交换后的结果。

【提示】将一个二维数组的行元素和列元素互换, 若一个一个去换, 则很麻烦, 可以利用循环来交换数组的行和列元素。

【核心代码】

```
for(i=0;i<4;i++)  
for(j=0;j<5;j++)  
y[j][i]=x[i][j];  
for(i=0;i<5;i++)  
for(j=0;j<4;j++)  
printf("%d ",y[i][j]);
```

6. 编写一个程序, 实现两个字符串的合并。

【提示】要实现两个字符串的合并, 可以先用指针移动至一个字符串的末尾, 然后将另一个字符串复制到字符串的末尾。

【核心代码】

```
gets(s1);  
gets(s2);  
p1=s1;  
p2=s2;  
while(*p!= '\0')  
p1++;  
*p1++=*p2++;
```

7. 编写一个程序, 将字符串倒序并输出至屏幕。

【提示】将字符串倒序输出, 可以利用指针先移动到字符串的末尾, 再通过指针实现字符串的倒序输出。

【核心代码】

```
gets(s);  
p=s;  
while(*p!= '\0')  
p++;  
while(p!=s)  
{  
printf("%c",*p);  
p--;  
}
```

8. 从键盘输入 10 个数据, 利用冒泡排序法对其从小到大排序后输出至屏幕。

【提示】将从键盘输入的 10 个数据保存至数组中, 利用冒泡排序法对其从小到大排序后输



出至屏幕。

【核心代码】

```
for(i=0;i<10;i++)
scanf("%d",&a[i]);
for(i=0;i<9;i++)
for(j=i+1;j<10;j++)
if(a[i]>a[j])
{
    t=a[i];
    a[i]=a[j];
    a[j]=t;
}
```

9. 从键盘输入一串字符，以#为结束标志，统计其中字符的个数。

【提示】从键盘输入的字符串，可以将其保存至字符数组中。然后逐个字符与字符#比较，若不相等则计数器加 1，若相等则停止计数。

【核心代码】

```
scanf("%s",s);
while(s[i]!='#')
{
    i++;
    count++;
}
```

10. 现有一字符串，删除特定位置上的字符后，输出至屏幕。

【提示】定义字符数组，用来保存字符串，然后移动到要删除的字符位置上，将后面的字符都向前移动一位，即可实现字符的删除功能。

【核心代码】

```
char s[]="Hello World! ";
for(i=n;i<=strlen(s);i++)
s[i]=s[i+1];
printf("%s",s);
```

第 8 章 指 针

指针是 C 语言中最重要的部分之一，也是最难掌握的。指针的概念很复杂，通过指针可以直接对地址操作，但很容易出错。正确地使用指针可以使程序的执行效率变高，并且变得简洁、高效。在本章中将讲解指针的概念及其应用，其中包括变量指针、数组指针、函数指针等。指针可以说是 C 语言的精华，在 C 语言的学习过程中起着至关重要的作用，因此要学好 C 语言，就必须认真学习和掌握 C 语言中的指针。

本章主要涉及的知识点有：

- 指针的概念；
- 数组指针；
- 变量指针；
- 函数指针；
- 指针的应用。



8.1 指针简介

指针是 C 语言的一大重点，利用指针可以直接对地址进行操作，可以快速高效的访问内存中的数据。在讲解指针之前，先讲解一下计算机的内存地址。因为指针是对内存地址进行操作的，要合理利用好指针就必须先了解计算机内存地址结构。

计算机中的内存被划分为一个个存储单元，其中每个存储单元是以字节为单位的。每一个存储单元都有自己的编号，计算机通过编号可以访问相应存储单元中的内容，这个编号就称为内存地址。编译系统会根据变量的类型所占的位数分配一定长度的存储空间，用来存放数据。例如有以下语句：

```
int x=1,y=2,z=3;
```

编译程序时，会给变量 x, y, z 分别分配大小为 2 个字节的存储空间。设其首地址为 3000，则其内存分配图如图 8-1 所示。

通过变量 x 可以直接找到内存地址 3000 中的内容即 1，可以直接对该地址进行操作例如赋值操作。这种方式称为直接访问。

在 C 语言中还有另外一种访存方式，即间接访问。间接访问就好像你要找一个人，你不知道他在哪，但你知道有一个人知道他在哪个地方。你可以通过那个人来找到你要找的人，指针就如同那个人一样，是一种媒介，可以找到变量的内存地址并对其进行操作。例如下面的语句：

```
int x=1,y=2,*p=x;
```

内存地址编号	内存内容
3000	1
3002	2
3004	3

图 8-1 内存地址分配图



定义两个整型变量 x , y , 另外定义了一个整型指针 p , 它是指向变量 x 的, 利用该指针可以对变量 x 内存地址中的内容进行操作。

其内存地址表如图 8-2 所示。

指针可以指向变量的内存地址, 并对其中的内容进行操作, 可以方便快捷的对数据进行操作。

内存地址编号	地址内容
3000	1
3002	2
4000	值不确定

图 8-2 内存地址表示图



8.2 指针的定义及应用

指针变量用来存放变量的地址值, 通过指针可以对变量进行操作。不同类型的变量占据的存储空间不同, 因此在 C 语言中有着不同类型的指针变量用来对变量进行操作。在本节中将重点讲解指针的定义及其使用。

8.2.1 指针的定义

要使用指针之前, 必须要先定义指针。其定义形式如下:

类型 *指针变量名;

其中*表示该变量为指针类型, 类型可以为任意的类型。在指针的使用过程中, 不同的数据类型应定义相应类型的指针。例如:

```
int *p;
```

定义了一个指向整型数据的指针变量 p , 可以用来存放整型变量的地址。指针若保存了整型变量的地址, 可对该地址中内容直接进行操作。例如:

```
int i=1;
int *p;
p=&i;
*p=3;
```

上述程序中, 声明整型指针变量指向整型变量 i 。利用 $*p=3$ 对整型变量 i 进行赋值, 它与 $i=3$ 执行的效果是一样的。

在使用指针时应注意以下两个方面:

(1) 声明指针变量时, 指针变量前一定要有 “*” 符号。(2) 定义指针时, 要加上类型标识符, 表明指针的类型。

8.2.2 指针的引用

C 语言中通过两个地址操作符可直接对内存地址中的内容进行操作, 即 “&” 和 “*”。 “&” 为取地址符, 可以用来获取一个变量的地址。 “*” 可以用于获取指针指向的地址中的内容, 并可以对该地址中的内容进行操作, 如赋值等。

指针变量定义后可给指针变量赋值, 例如:

```
p=&a;
```

表示将变量 `a` 的地址赋给 `p`，即使指针 `p` 指向变量 `a` 的内存地址。
指针变量赋值后，还可以改变变量的值。例如：

```
int i;
int *p=&i;
*p=3;
```

【范例 8.1】 编写程序，通过指针访问变量并输出变量的值。

分析：指针访问变量，必须要首先将变量的地址赋给指针，使指针指向变量的存储地址。这样指针才可以访问变量的值并将其输出。

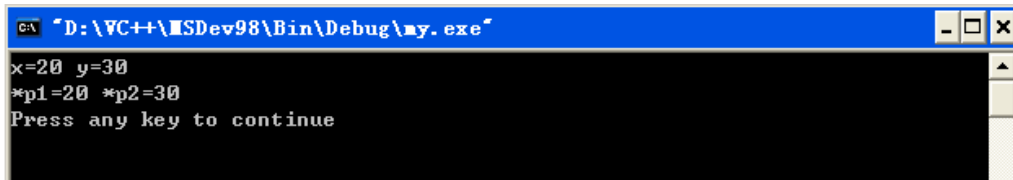
范例 8.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int x,y;                                        /*定义整型变量 x 和 y*/
05      int *p1,*p2;                                    /*定义整型指针 p1 和 p2*/
06      x=20,y=30;
07      p1=&x;                                          /*使指针 p1 指向变量 x*/
08      p2=&y;
09      printf("x=%d y=%d\n",x,y);
10      printf("*p1=%d *p2=%d\n",*p1,*p2);
11  }
```

【代码分析】 本例是指针应用的简单范例，详细代码分析如下：

- 第 5 行，定义两个整型指针变量 `p1` 和 `p2`，但它们没有指向任何整型变量。
- 第 7、8 行，将 `x` 和 `y` 的内存地址分别赋给指针 `p1` 和 `p2`。
- 第 9、10 行，输出变量 `x`，`y` 和指针 `p1`，`p2` 的值，比较其值是否相同。

【运行结果】 该程序的执行结果如图 8-3 所示。



```
C:\ "D:\VC++\MSDev98\Bin\Debug\my.exe"
x=20 y=30
*p1=20 *p2=30
Press any key to continue
```

图 8-3 范例 8.1 结果图



注意：用指针输出数据时，要在指针前加上“*”号。若不加上“*”号，输出的是变量的地址，只有加上“*”号，输出的才是变量地址中的内容。

在使用地址操作符“&”和“*”应注意以下两个方面：

(1) 若有 `p1=&x`；`p2=&*p1`；则 `p2` 的值为 `&x` 即 `&*p1` 等价于 `&x`。因为 `&` 和 `*` 是按从右至左的规则结合的，`*p1` 的值为 `x`，再对其取地址即为 `&x`。因此指针 `p1` 和 `p2` 指向同一地址，即变



量 x 的指针。

(2) $(*p1)++$ 与 $x++$ 是等价的。 $(*p1)++$ 是指针 $p1$ 指向的变量加上 1，因此与 $x++$ 执行的结果是一样的。



提示： $(*p1)++$ 与 $*p1++$ 是不一样的。 $(*p1)++$ 表示指针指向的变量值加 1， $*p1++$ 表示指针指向的变量地址加上 1，再获取地址中的内容。

【范例 8.2】 从键盘输入两个数 x 和 y，将这两个数从小到大输出。

分析：比较两个数 x 和 y 的大小，可用直接访问的方式比较两个数的大小，若变量 x 大于 y，则交换两个数的值输出。另外也可以使用指针 $p1$ 和 $p2$ 分别指向 x，y，再对其进行比较，若 x 大于 y 交换指针 $p1$ ， $p2$ 的指向，最后输出指针 $p1$ 和 $p2$ 的值。

范例 8.2 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x,y;
05      int *p1,*p2,*p3;
06      scanf("%d%d",&x,&y);          /*调用 scanf()函数实现数据的输入*/
07      p1=&x;
08      p2=&y;
09      if(x>y)                      /*若 x 大于 y，则交换 p1 和 p2 的指向*/
10      { p3=p1;p1=p2;p2=p3;}
11      printf("x=%d y=%d\n",x,y);    /*输出变量 x 和 y 的值*/
12      printf("*p1=%d *p2=%d\n",*p1,*p2);
13  }
```

【代码分析】 本例利用指针实现交换数据，详细代码分析如下：

- 第 9、10 行，比较 x 和 y 的大小。若 x 大于 y 则交换 $p1$ 和 $p2$ 的指向，即 $p1$ 指向 y， $p2$ 指向 x。

【运行结果】 该程序的执行结果如图 8-4 所示。

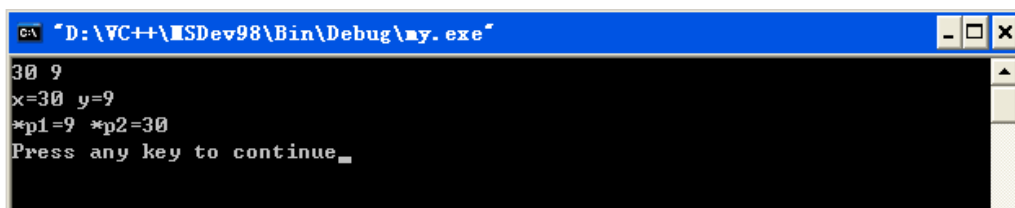


图 8-4 范例 8.2 结果图

上例中指针 $p1$ 和 $p2$ 的指向变化如图 8-5 所示。



注意： $p1$ 和 $p2$ 的值虽然改变了，但 x 和 y 的值并未改变。

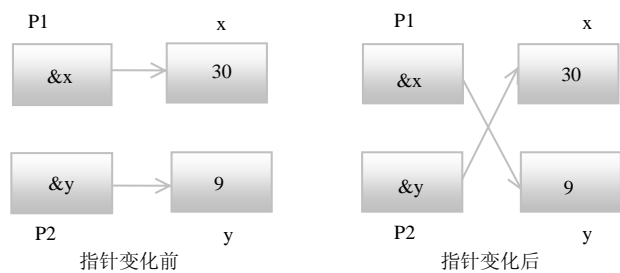


图 8-5 指针变量变化情况

8.2.3 指针变量作为函数的参数

数组名可以作为函数的参数给地址传递一个函数，指针保存变量的地址值，因此也可以作为参数传递给函数。下面通过一个实例来讲解指针如何作为函数的参数。

【范例 8.3】从键盘输入两个数，利用指针交换两个数后输出。

分析：通过 `scanf()` 函数从键盘获取两个数保存至变量 `x` 和 `y` 中，定义两个指针变量 `p1` 和 `p2` 分别指向 `x` 和 `y`。自定义一个函数，用来交换 `x` 和 `y` 的值，其中 `p1` 和 `p2` 作为参数传递给自定义函数。利用自定义函数交换 `x` 和 `y` 的值，然后输出 `x` 和 `y` 的值。

范例 8.3 代码实现

```
01  #include <stdio.h>
02  void change(int *s1,int *s2)          /*自定义函数 change()*/
03  { int t;
04    t=*s1;                              /*通过变量 t 交换*s1 和*s2 的值*/
05    *s1=*s2;
06    *s2=t;
07  }
08  void main()
09  {
10    int x,y;
11    int *p1,*p2;
12    scanf("%d%d",&x,&y);
13    p1=&x;p2=&y;
14    printf("x=%d,y=%d\n",x,y);          /*输出交换前变量 x 和 y 的值*/
15    change(p1,p2);                      /*调用函数 change(),交换变量 x 和 y 的值*/
16    printf("x=%d,y=%d\n",x,y);          /*输出交换后变量 x 和 y 的值*/
17  }
```

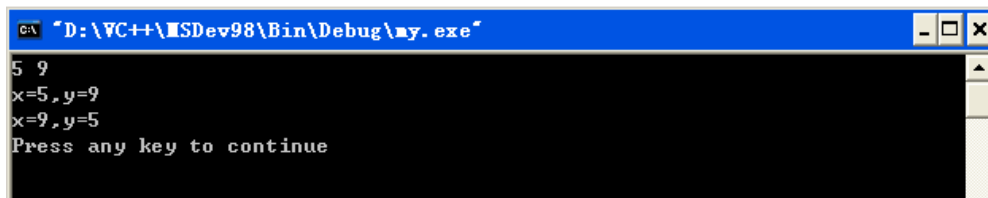
【代码分析】本例利用指针作为函数参数传递，详细代码分析如下：

- 第 2~7 行为自定义函数 `change()`，其中指针 `s1` 和 `s2` 用来接受 `p1` 和 `p2` 的地址值。在 C 语言中，`s1` 和 `s2` 称做形式参数，简称形参，`p1` 和 `p2` 称为实际参数，简称实参，详细内容将会在后面的章节中讲解到。
- 第 15 行，调用自定义函数 `change()` 实现两个数的交换，`p1` 和 `p2` 作为实参传递给函数



的形参 s1 和 s2。

【运行结果】该程序的执行结果如图 8-6 所示。



```
C:\ "D:\VC++\SDev98\Bin\Debug\my.exe"
5 9
x=5, y=9
x=9, y=5
Press any key to continue
```

图 8-6 范例 8.3 结果图



注意：change()函数若改成以下形式：

```
change(int s1,int s2)
{ int t;
  t=s1;
  s1=s2;
  s2=t;
}
```

在主函数 main()函数中调用 change()函数将不能实现两个数的对换。因为 s1 和 s2 是 change()函数的形式参数，它们只在函数内起作用，对函数外的 x 和 y 不会造成影响。而*s1 和*s2 是接受了 x 和 y 的地址，然后交换 x 和 y 地址中的值，因此可以交换 x 和 y 的值。



8.3 指针与数组

数组是相同类型元素的集合，在内存中占据着一块连续的存储空间，每个元素都有一个确定的地址值。因此可以利用指针对数组中的每一个元素进行操作，在本节中将讲述如何利用指针对数组进行操作，其中包括一维数组和多维数组。

8.3.1 指针和一维数组

在 C 语言中，数组名可表示数组的首地址，即数组第一个元素所在的位置。在数组中通过下标可以访问数组中的元素，定义指向数组元素的指针也可以实现访问数组元素的功能。例如：

```
int x[5];
int *p=a;
```

定义了一个指向整型数组 x 的指针变量 p，其中*p=a 与*p=&a[0]是等价的，因为数组名可代表数组首地址，其功能都是将数组首地址赋给指针 p。

在 C 语言中，若指针 p 指向数组中某一元素，则 p+1 指向数组的下一个元素。p=x+1 等价于&x[1]，即将第 2 个元素赋给指针 p。p=x+i 可将第 i+1 个元素赋给指针 p，等价于&x[i]。

访问数组中一个元素可以有三种方法：



(1) 通过下标访问: $x[i]$ 。(2) 通过地址访问: $*(x+i)$ 。(3) 通过指针访问: $*(p+i)$ 。
上述三种方法都是等价的, 都可实现对数组元素的访问。

【范例 8.4】 利用上述三种方法, 输出整型数组 $x[5]$ 中每个元素的值。

分析: 输出数组 x 中的元素, 可以利用下标、地址、指针来访问。若用下标法来访问, 可以利用循环输出 $x[i]$ 。若用地址访问, 输出 $*(a+i)$ 的值即可。若用指针访问, 先使指针指向数组 x , 再输出 $*(p+i)$ 即可得到数组中的每个元素。

范例 8.4 代码实现

(1) 通过下标访问数组 x 。

```
01 void main()  
02 {  
03     int x[5];  
04     int i;  
05     for(i=0;i<5;i++)  
06         scanf("%d",&x[i]);  
07     for(i=0;i<5;i++)          /*通过 for 循环和 printf() 函数输出数组中的每个元素*/  
08         printf("%d ",x[i]);  
09     printf("\n");  
10 }
```

(2) 通过地址访问数组 x 。

```
01 void main()  
02 {  
03     int x[5];  
04     int i;  
05     for(i=0;i<5;i++)          /*通过 for 循环和 scanf() 函数实现数据的输入*/  
06         scanf("%d",&x[i]);  
07     for(i=0;i<5;i++)          /*通过地址访问法输出数组中每个元素*/  
08         printf("%d ",*(x+i));  
09     printf("\n");  
10 }
```

(3) 通过指针访问数组 x 。

```
01 void main()  
02 {  
03     int x[5];  
04     int i,*p;  
05     for(i=0;i<5;i++)  
06         scanf("%d",&x[i]);  
07     for(p=x;p<(x+5);p++)      /*通过指针访问并输出数组中的元素*/  
08         printf("%d ",*p);  
09     printf("\n");  
10 }
```



【代码分析】本例利用三种方法访问数组 `x`，详细代码分析如下：

- 第 5、6 行，三种方法都是利用 `for` 循环和 `scanf()` 函数实现数组的输入。
- 第 7、8 行，利用 `for` 循环以及不同的方法输出数组中的各个元素。

【运行结果】该程序的执行结果如图 8-7 所示。

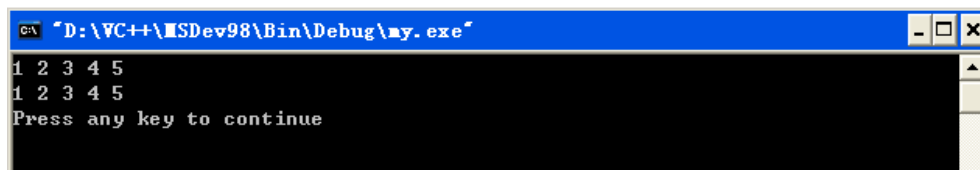


图 8-7 范例 8.4 结果图

在上述三种方法中，通过指针访问数组的执行效率最高。指针访问往往不直观，很难理解，而通过下标引用数组元素最为直观。

设 `p` 为指针变量，`x` 为一维数组，`p=x` 则应注意以下三个方面：

(1) 指针 `p` 的值改变则其指向也会改变，引用的数组元素也将不同。例如，`p--` 使 `p` 指针指向 `x` 数组中上一个元素。

(2) 指针不应越界，否则结果会出错。例如以下程序：

```
void main()
{
    int x[5];
    int i, *p=x;
    for(i=0; i<5; i++; p++)
        scanf("%d", p);
    for(p=x; p<x+5; p++)
        printf("%d ", *p++);
}
```

上述程序中 `p` 很明显越界了，超过了数组下标的范围，因此程序运行结果将会出错。

(3) 注意++、--运算符的区别。

- `*p++` 是先取出 `*p` 的值，再使 `p` 加 1。
- `*p--` 是先取出 `*p` 的值，再使 `p` 减 1。
- `(*p)++` 是使 `*p` 的值加 1。
- `(*p)--` 是使 `*p` 的值减 1。
- `*(p++)` 等价于 `*p++`，先取出 `*p` 的值，再使 `p` 加 1。
- `*(p--)` 等价于 `*p--`，先取出 `*p` 的值，再使 `p` 减 1。

例如以下程序：

```
void main()
{
    int x=3;
    int *p, *q;
    p=&x; q=&x;
    printf("%d\n", *p++);           /*输出*p++的值*/
}
```

```
printf("%d\n",(*q)++);
printf("%d",x);
}
```

其运行结果如图 8-8 所示。

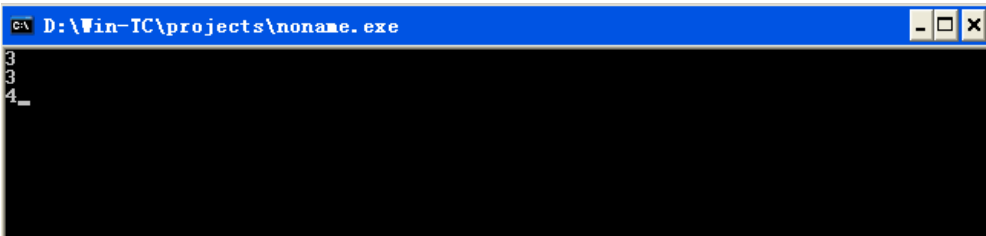


图 8-8 运行结果图

8.3.2 指针和二维数组

指针不仅可以引用一维数组，还可以引用多维数组中的元素。多维数组是指超过一维的数组，二维数组在 C 语言中使用较多，三维以上数组由于其复杂性一般不使用。假设有以下二维数组：

```
int x[2][4]={0,1,2,3,4,5,6,7};
```

这里定义了一个二维数组 x，其中总共有 8 个元素，分别为 x[0][0]、x[0][1]、x[0][2]、x[0][3]、x[1][0]、x[1][1]、x[1][2]、x[1][3]。

在二维数组 x 中，x 表示二维数组的首地址即&x[0][0]，x+1 表示第 1 行的首地址即&x[1][0]。若 x 的首地址为 2000，因为二维数组每行有 4 个元素，因此 x+1 的地址值为 2008。

根据数组表示规则，二维数组 x 每个元素地址可表示为：

x[0]+0	x[0]+1	x[0]+2	x[0]+3
x[1]+0	x[1]+1	x[1]+2	x[1]+3

等价为：

&a[0][0]	&a[0][1]	&a[0][2]	&a[0][3]
&a[1][0]	&a[1][1]	&a[1][2]	&a[1][3]

【范例 8.5】通过指针输出二维数组每个元素的值。

分析：定义一个指针变量指向二维数组的首地址，再根据二维数组的大小利用 for 循环和指针进行相应的输出。

范例 8.5 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[2][4]={0,1,2,3,4,5,6,7};          /*定义二维数组 x 并进行初始化*/
05      int *p;
```




```
06   for(p=x[0];p<x[0]+8;p++)           /*通过指针变量来控制 for 循环的执行*/
07   {
08       if((p-x[0])%4==0)               /*若已输出 4 个元素则换行*/
09       printf("\n");
10       printf("%d ",*p);
11   }
12   printf("\n");
13 }
```

【代码分析】本题通过指针访问二维数组 `x`，详细代码分析如下：

- 第 5 行，定义了整型指针变量，但没有对该指针赋值。它可以指向整型变量，也可以指向整型数组。
- 第 6 行，`p=x[0]` 执行后，指针 `p` 指向二维数组 `x` 的首地址，`p++` 可使 `p` 指向数组中的下一个元素。
- 第 8、9 行，用 `if` 语句判断，若一行中已输出 4 个元素则换行。

【运行结果】该程序的执行结果如图 8-9 所示。

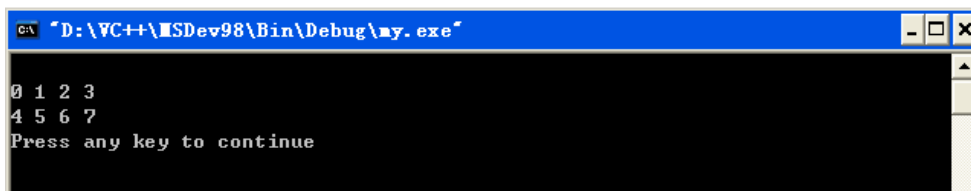


图 8-9 范例 8.5 结果图

访问二维数组可以直接使用指针变量，也可以通过执行 `n` 个元素组成的一维数组的指针来引用各个元素。其定义形式如下：

类型 (*指针名)[`n`];

例如：

```
int (*p)[5];
```

定义了一个指向含有 5 个元素组成的一维数组的指针变量 `p`，其中圆括号一定不能去掉。下面通过一个例子来看下指向一维数组的指针变量的使用。

【范例 8.6】从键盘输入一个二维数组，利用指针输出该数组的任意一个元素。

分析：利用 `scanf()` 函数和 `for` 循环实现数组的输入，同时定义一个指针变量指向二维数组的首地址。因为要利用指针输出数组的任意一个元素，因此要从键盘输入要输出的元素所在的行和列，再利用指针输出该行该列的元素。

范例 8.6 代码实现

```
01   #include <stdio.h>
02   void main()
03   {
04       int x[2][4];
```



```

05  int i,j,(*p)[4];
06  for(i=0;i<2;i++)
07      scanf("%d%d%d%d",&x[i][0],&x[i][1],&x[i][2],&x[i][3]);
08  p=x;                                /*使指针数组p指向数组x*/
09  printf("input i and j: ");
10  scanf("%d%d",&i,&j);
11  printf("x[%d][%d]=%d\n",i-1,j-1,*(*(p+i-1)+j-1)); /*输出 x[i-1][j-1]的值*/
12  }

```

【代码分析】本例通过指针访问二维数组 x 中任意元素，详细代码分析如下：

- 第5行，定义两个整型变量 i 、 j 和指针变量 p 。其中 i 和 j 用于确定要输入元素所在的行和列。指针 p 定义为指向包含4个元素的一维数组的指针。
- 第9、10行，输入要输出的元素所在的行和列值。

【运行结果】该程序的执行结果如图8-10所示。

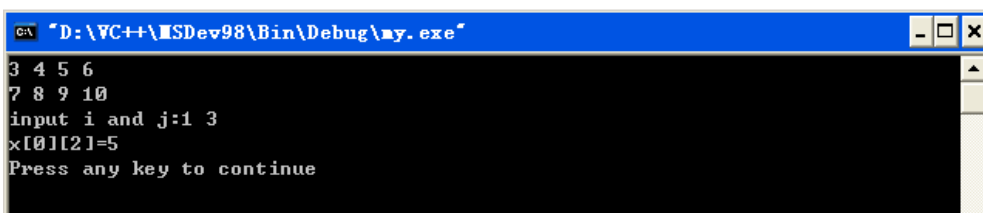


图8-10 范例8.6结果图



注意：第5行中 $(*p)[4]$ 的圆括号不能省去，若写出 $*p[4]$ 则表示完全不同的意思。
 $(*p)[4]$ 是声明一个指向包含4个元素的一维数组的指针，而 $*p[4]$ 表示声明一个长度为4的指针数组 p 。



8.4 指针和字符串

在C语言中，用字符数组来存储字符串。通过前面的学习可知道，定义一个指针指向用来存放字符串的字符数组，就可通过指针运算对字符串进行各种操作。

一个字符串若被赋给一个字符型数组，则数组名可表示字符串的首地址，将字符指针指向该字符数组即可对字符串进行操作。例如：

```
char *p="abcd";
```

定义了字符指针 p ，并将字符串“abcd”的第1个字符 a 的首地址赋给指针 p 。

在使用字符数组和指针时应注意以下两个方面：

(1) 字符数组名为地址常量，不能改变其值。例如：

```
char s[10]= "abcdef";
```

若改成：



```
char s[10];  
s="abcdef";
```

是错误的。s 是字符数组名，表示数组首地址，不能重新赋值。

(2) 可以直接将字符串常量赋值给指针。例如：

```
char *s="abcdef";
```

也可改写为：

```
char *s;  
s="abcdef";
```

上述两种写法都正确，都是将字符串“abcdef”赋值给指针变量 s。其中 s 指向字符串“abcdef”的首地址，即字符 a，*(s+1)为字符 b，*(s+2)为字符 c，*(s+3)为字符 d。通过指针变量 s 可以改变字符串的内容，即重新赋值。

【范例 8.7】 编写程序，利用指针改变字符串的内容输出至屏幕。

分析：在 C 语言中，字符数组可以用来存储字符串。但字符数组的大小必须要大于字符串的长度，当不知道字符串的长度时，为了避免越界，则应定义一个足够大的字符数组用来存放字符串。这样做会造成存储空间的浪费，而指针是存储字符串首地址的，利用指针可以克服这一问题。

范例 8.7 代码实现

```
01  #include <stdio.h>  
02  void main()  
03  {  
04      char *p="Hello C! ";  
05      puts(p);                               /*输出该字符串至屏幕*/  
06      p="Hello everybody! ";                 /*给指针 p 进行重新赋值*/  
07      puts(p);  
08  }
```

【代码分析】 本例通过指针改变字符串内容，详细代码分析如下：

- 第 4 行，定义了一个字符指针 p，将字符串“Hello C!”赋值给指针 p。
- 第 5 行，利用 puts() 函数输出字符指针 p 中的内容。
- 第 6 行，对指针 p 重新赋值，将字符串“Hello everybody!”赋值给指针 p。

【运行结果】 该程序的执行结果如图 8-11 所示。

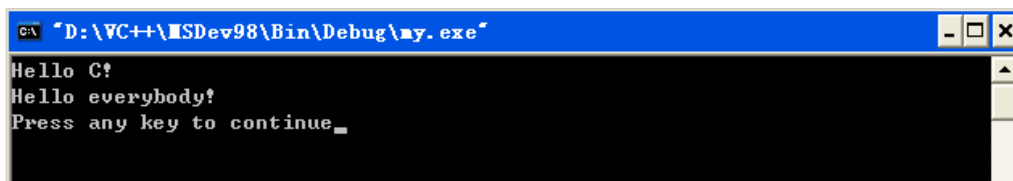


图 8-11 范例 8.7 结果图



【范例 8.8】编写一个程序，实现从键盘输入两个字符串 a1 和 a2，将字符串 a2 连接到字符串 a1 的后面，要求不利用库函数 strcat()。

分析：定义两个字符指针 p1 和 p2，使指针 p1 指向字符串 a1 的首地址，指针 p2 指向字符串 a2。利用循环使指针 p1 指向字符串 a1 的末尾，再利用指针 p1 和 p2 逐位移动，将字符串 a2 的内容连接到字符串 a1 的后面，最后在 a1 的末尾加上结束标识符\0 即可实现两个字符串的合并。

范例 8.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      char a1[50],a2[20];
05      char *p1=a1,*p2=a2;
06      printf("input a1: ");
07      gets(a1);                /*通过 gets() 函数从键盘获取字符至数组 a1 中*/
08      printf("input a2: ");
09      gets(a2);
10      while(*p1)                /*将指针 p1 移动到字符串末尾*/
11          p1++;
12      /*通过 while 循环将另一个字符串连接到该字符串之后*/
13      while(*p2)
14          *p1++=*p2++;
15      *p1='\0';                /*加上'\0' 结束标识符*/
16      printf("the String is :%s\n",a1);
17  }
```

【代码分析】本例通过指针实现合并字符串，详细代码分析如下：

- 第 5 行，定义两个字符指针 p1 和 p2，分别指针数组 a1 和 a2 的首地址。
- 第 6~9 行，利用 gets() 函数实现字符串 a1 和 a2 的输入。
- 第 10~14 行为字符串的合并过程。先使指向字符串 a1 的指针 p1 移至末尾，然后将字符串 a2 中的内容复制到 a1 的后面，最后在 a1 的后面加上结束标识符\0。

【运行结果】该程序的执行结果如图 8-12 所示。

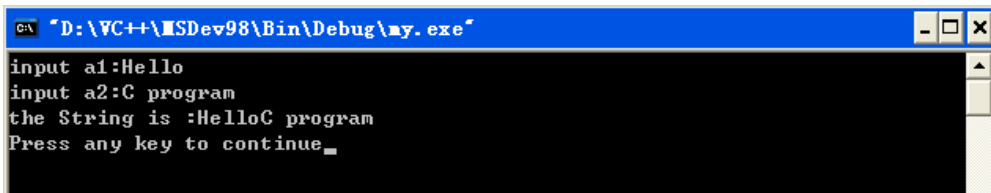


图 8-12 范例 8.8 结果图



8.5 函数的指针

在 C 语言中，变量、数组等都有地址，可用指针来操作。同样函数也有地址，也可以通过



指针来进行操作。每一个函数都会有一个入口地址，通过该地址可以调用函数。

函数的指针定义形式如下：

```
int (*f)();
```

声明一个指向函数的指针变量 *f*。声明了函数的指针变量后，就可以通过指针调用函数。下面通过一个例子了解函数指针的使用。

【范例 8.9】从键盘输入两个数 *x* 和 *y*，输出两个数中较大的数。

分析：编写一个函数进行两个数的比较，将 *x* 和 *y* 作为参数传递给函数比较大小，返回两个数中的大者。定义一个函数的指针，使指针指向函数的入口地址，通过指针来调用函数比较两个数的大小，最后输出较大的数。

范例 8.9 代码实现

```
01  #include <stdio.h>
02  void main()
03  { int max();                      /*声明自定义函数 max()*/
04    int x,y,z,(*p)()=max;          /*定义变量 x,y,z 及函数指针 p*/
05    scanf("%d%d",&x,&y);
06    z=(*p)(x,y);                   /*通过指针 p 调用函数比较变量 x 和 y 大小*/
07    printf("the max is %d",z);
08    getch();
09  }
10  int max(int a,int b)              /*自定义函数 max()*/
11  { int c;
12    c=a>b?a:b;                     /*通过条件运算符将 a 和 b 中大者赋给变量 c*/
13    return c;                      /*返回变量 c 的值*/
14  }
```

【代码分析】本例通过指针调用函数，详细代码分析如下：

- 第 3 行为函数声明。在调用函数之前必须先声明除非在调用之前已给出函数代码。
- 第 4 行，定义了 3 个整型变量和一个函数指针 *p*。其中 *x* 和 *y* 用于保存输入的整数，*z* 用于保存两个数比较大者，函数指针 *p* 指向 *max()* 函数的入口地址。
- 第 6 行，通过函数指针 *p* 调用 *max()* 函数。
- 第 10~14 行，自定义的 *max()* 函数。其中 *a*、*b* 为函数的形参，它们的值与传入函数的实参 *x* 和 *y* 的值相等，比较 *a* 和 *b* 的值后返回两者中较大的数。

【运行结果】该程序的执行结果如图 8-13 所示。

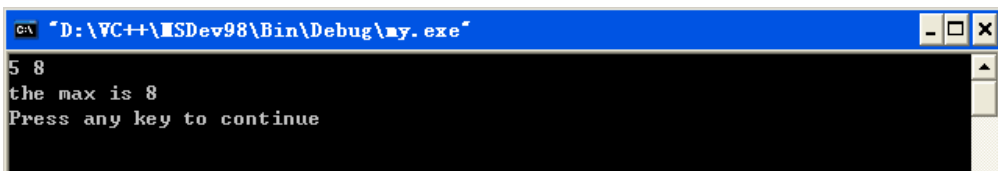


图 8-13 范例 8.9 结果图



注意：

(1) 函数的指针变量不一定指向某一个特定的函数，可以指向任意函数。当要改变指针要调用的函数时，只需要将函数的入口地址赋给指针即可。(2) 将函数的入口地址赋给函数的指针时只需将函数名赋给指针，不用带参数。(3) 当用函数的指针调用函数时，用(*p)表示函数名，仍然要带实参，例如(*p)(x,y)。(4) 指向函数指针的值，不应随意改变，否则不能调用相应的函数。



8.6 指向指针的指针

在前面讲解的指针中，都是直接指向变量的指针。在 C 语言中，指针还可以指向指针。若一个指针不是指向变量，而是指向另外一个指针，前一个指针才指向变量的地址。这种指针称为指向指针的指针，也称为二级指针。其形式如图 8-14 所示。



图 8-14 二级指针访问形式

二级指针的定义形式如下：

类型 **指针名；

例如：

```
char **p;
```

定义了一个字符型的二级指针，下面通过一个实例来了解二级指针的应用。

【范例 8.10】编写程序，通过二级指针输出多个字符串。

分析：利用一维指针数组存储多个字符串，定义一个二级指针指向该指针数组的首地址，通过对地址进行操作输出字符串。

范例 8.10 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i;
05      char **p;                                /*定义二级指针 p*/
06      char *s[]={ "Hello", "student", "teacher", "team" }; /*定义字符指针数组 s*/
07      p=s;                                     /*使二级指针 p 指向字符指针数组 s*/
08      for(i=0;i<4;i++)
09          printf("%s\n",p[i]);
10      for(i=0;i<4;i++)                        /*输出前 4 个字符元素的首地址*/
```



```
11     printf("%d\n", (*p)++);
12 }
```

【代码分析】本例为二级指针应用范例，详细代码分析如下：

- 第 6 行，定义一个一维指针数组，并对其进行初始化。
- 第 7 行，将数组 *s* 的首地址赋给二级指针 *p*，即使二级指针 *p* 指向指针数组 *s*。
- 第 8、9 行，利用二级指针输出数组 *s* 中的每个字符串。
- 第 10、11 行，二级指针 *p* 指向数组 *s* 的首地址，因此 $(*p)++$ 指向第 1 个字符串中每个字符的地址。设第 1 个字符串的首地址为 412，则后面的结果为 413，414，415。

【运行结果】该程序的执行结果如图 8-15 所示。

```
C:\ "D:\VC++\MSDev98\Bin\Debug\my.exe"
Hello
student
teacher
team
4337728
4337729
4337730
4337731
Press any key to continue
```

图 8-15 范例 8.10 结果图



8.7 指针应用举例

【范例 8.11】现已有 5 个给定的字符串，将其按从小到大的顺序排列并输出。

分析：对多个字符串的存储，可以采用字符型二维数组或指针数组来实现。字符串大小的比较可以直接调用库函数 `strcmp()` 函数实现。

范例 8.11 代码实现

```
01  #include <stdio.h>
02  #include <string.h>                /*包含 string.h 头文件*/
03  void main()
04  {
05      int i,j;
06      char *p;
07      char *s[]={ "Hello", "student", "teacher", "team", "school" };
08      /*通过 for 循环和 strcmp() 函数比较数组 s 中每个元素的大小并进行排序*/
09      for(i=0; i<4; i++)
10          for(j=i+1; j<5; j++)
11              if(strcmp(s[i], s[j])>0)                /*若 s[i] 大于 s[j] 则交换这两个元素*/
12                  { p=s[i];
13                    s[i]=s[j];
14                    s[j]=p;
15                }
```



```

14     }
15     for(i=0;i<5;i++)
16         printf("%s\n",s[i]);
17     }

```

【代码分析】本例为字符串比较范例，详细代码分析如下：

- 第2行，将 string.h 头文件包含进来。因为在下面的程序中将用到该文件中的 strcmp() 函数，因此必须先将该头文件包含进来。
- 第8~14行，利用冒泡排序法对字符串进行排序，其中字符串大小的比较是利用库函数 strcmp() 来实现的。

【运行结果】该程序的执行结果如图 8-16 所示。

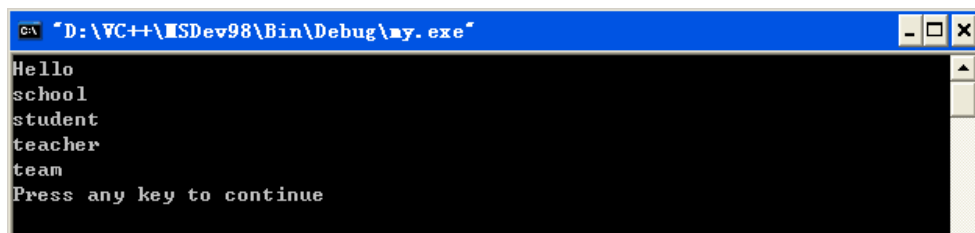


图 8-16 范例 8.11 结果图

【范例 8.12】现有 5 名学生，4 门科目：语文、数学、英语、化学。考试之后要对这 5 名学生做以下的统计：

- (1) 求第一门课语文科目的平均成绩。
- (2) 求每一名学生的平均成绩并将其输出。

分析：5 名学生 4 门科目，因此可以定义一个大小为 5×4 的二维数组来保存成绩。学生的成绩不一定为整数，数组应定义为浮点型。另外编写两个函数用来计算语文科目的平均成绩和每一名学生的平均成绩，最后在主函数中调用这两个函数并输出结果。

范例 8.12 代码实现

```

01  #include <stdio.h>
02  void avcourse(float *p1)                /*自定义函数 avcourse()*/
03  {
04      int i;
05      float sum=0,average;
06      for(i=0;i<5;i++)                    /*统计第一门科目的总成绩*/
07          sum=sum+(p1+4*i);
08      average=sum/5;                      /*将总成绩除以人数求得平均成绩*/
09      printf("course 1 average is %f\n",average);
10  }
11  void aver(float *p1)                    /*自定义函数 aver()*/
12  {
13      int i,j;
14      float sum,average;
15      for(i=0;i<5;i++)

```




```
16 { sum=0;
17   for(j=0;j<4;j++)           /*计算每名学生成绩的总和*/
18     sum=sum+*(p1+4*i+j);
19   average=sum/4;             /*求每名学生的平均成绩*/
20   printf("student %d:average is %f\n",i+1,average);
21 }
22 }
23 void main()
24 {
25   int i;
26   float score[5][4], *p1,*p2;
27   for(i=0;i<5;i++)
28   {
29     printf("input student %d score\n",i+1);
30     scanf("%f%f%f%f",&score[i][0],&score[i][1],&score[i][2],&score[i][3]);
31   }
32   p1=&score[0][0];
33   avcourse(p1);               /*调用 avcourse() 函数求第一门科目的平均成绩*/
34   aver(p1);                   /*调用 aver() 函数求每名学生的平均成绩*/
35 }
```

【代码分析】本例为字符串比较范例，详细代码分析如下：

- 第 2~10 行为自定义的 avcourse() 函数。该函数计算第一门科目语言的平均成绩，其中指针 p1 用来接收实参，即 score 数组的首地址。
- 第 11~22 行为自定义的 aver() 函数。此函数功能为求每一个学生的平均成绩并将计算得出的结果输出，指针 p1 也是用来接收 score 数组的首地址。
- 第 33~34 行，调用自定义的 avcourse() 和 aver() 函数计算平均分并输出。

【运行结果】该程序的执行结果如图 8-17 所示。

```
input student 1 score
80.5 78.5 88.0 90.0
input student 2 score
77.5 87.0 84.0 86.5
input student 3 score
79.0 89.0 90.0 91.0
input student 4 score
89.5 90.0 75.5 78.0
input student 5 score
69.0 90.5 88.5 87.0
course 1 average is 79.099998
student 1:average is 84.250000
student 2:average is 83.750000
student 3:average is 87.250000
student 4:average is 83.250000
student 5:average is 83.750000
```

图 8-17 范例 8.12 结果图



注意：自定义函数代码必须要在调用前就给出，若函数代码写在调用之后，则需在主函数中加上函数原型声明。



8.8 小结

指针是 C 语言中的难点，也是 C 语言的精髓。通过指针可以直接对地址进行操作，可以动态分配内存、可以方便使用数组和字符串，可以调用函数等。但指针很容易出现运算出错，造成程序出错并且难以发现。合理地利用指针，可以使程序更简洁，提高执行效率。因此指针对 C 语言的学习很重要，读者应认真掌握和研究指针的概念。



8.9 习题

一、选择题

1. 下列语句不正确的是 ()。

A. `int *p;`

B. `int *p[5];`

C. `int p(5)`

D. `int (*p)()`

【提示】A 选项中定义了一个整型变量的指针 p，B 选项是定义了一个整型指针数组，D 选项中定义了一个指向函数的指针 p。

2. 设有 `int x, y, z, *p=&x;`，则能实现从键盘输入 3 个数保存至变量 x, y, z 中的语句是 ()。

A. `scanf("%d%d%d", *p, y, z);`

B. `scanf("%d%d%d", p, y, z);`

C. `scanf("%d%d%d", &p, y, z);`

D. `scanf("%d%d%d", p, &y, &z);`

【提示】本题考察指针的使用，指针 p 指向变量 x，因此可以通过指针 p 对变量 x 进行操作。p 的值即为变量 x 的地址，调用 `scanf()` 函数是存储数据至变量的地址中。

3. 若有以下语句：`int x=10, *p=&x;`，则 `printf("%d\n", *p++);` 的结果为 ()。

A. 11

B. 10

C. 程序有误

D. 值不确定

【提示】本题考察指针的概念，很多人可能会误以为 `*p++` 的结果为 11，其实不对。在上述语句中，`*p=&x` 将变量 x 的地址赋给指针 p，即 `*p` 的值为 10。`printf("%d\n", *p++)` 语句中其中 `*p++` 是先取出 `*p`，再对 p 进行操作的，因此输出结果为 `*p` 的值即 10。

4. 对于数组 `x[8]`，下列不能表示数组元素 `x[2]` 地址的是 ()。

A. `&x[0]+2`

B. `&x[2]`

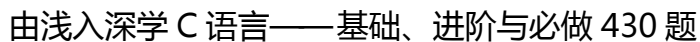
C. `x+2`

D. `&x[1]++`

【提示】本题考察地址运算符 & 的应用。数组名可表示数组的首地址，本题中定义一个整型数组 x，A、B、C 选项都能表示 `x[2]` 元素的地址。D 选项中 `&x[1]++` 是先取 `x[1]` 元素的地址，再对其进行加 1。因此 D 选项错误，它表示的是元素 `x[1]` 的地址。

5. 下列程序的运行结果为 ()。

```
void a(int *p)
{
```



A. 30
B. 31
C. 不确定
D. 程序有误

6. 设有语句 `char s[5]="abcd", *p;p=s;` 则 `* (p+1)` 的值为 ()。

7. 设有一语句: `int *p();` 则其表示 ()。

8. 现有以下程序，则其运行结果为（ ）。

A. 5
C. 13

B. 6
D. 12

9. 现有以下程序，其运行结果为（ ）。

188



```

while(*s)
{
    if(*s==temp)
        *s=temp-'c'+'B';
    s++;
}
}

void main()
{
    char *s1="abcdefafvg",c='f';
    m(s1,c);
    printf("%s",s1);
}

```

- A. abcdefafvg B. abcdeEaEvg
C. abcdeDaDvg D. abcdeavg

【提示】本题考察指针作为函数参数的概念。自定义函数 `m()` 中有两个参数，分别为字符型指针 `s` 和字符型变量 `temp`。在主函数定义一个字符指针变量和字符型变量，并对其初始化。将这两个变量作为参数调用 `m()` 函数，其功能为判断字符串中是否有字符 `f`，若有字符 `f`，则用 `E` 替换该字符。因此最后输出结果为 `abcdeEaEvg`。

10. 现有以下程序，则其运行结果为 ()。

```

int *s(int *a,int *b)
{
    if(*a<*b)
        return a;
    else
        return b;
}

void main()
{
    int a=5,b=9,*p1,*p2,*p3;
    p1=&a;
    p2=&b;
    p3=s(p1,p2);
    printf("%d,%d,%d",*p1,*p2,*p3);
}

```

- A. 5, 9, 9 B. 5, 9, 5
C. 9, 5, 9 D. 9, 5, 5

【提示】自定义一个函数 `s()`，其返回值为指针类型，其功能为比较两个数的大小并返回较小的数。在主函数中调用 `s()` 函数，`p1` 和 `p2` 作为函数的参数，并用指针 `p3` 接收函数的返回值。最后利用 `printf()` 函数输出指针 `p1`, `p2`, `p3` 的值，即 `a`, `b` 及两者中较小的数。

11. 以下程序运行结果为 ()。

```

void main()
{

```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
int x[5]={1,2,3,4,5};
int a,b,*p;
p=x;
a=*(p+1);
b=*(p+3);
printf("%d,%d",a,b);
}
```

A. 1, 3

B. 2, 4

C. 1, 4

D. 2, 5

【提示】上述程序将数组 x 第 2 个元素赋值给变量 a，数组 x 中第 4 个元素赋值给变量 b。

12. 以下程序运行结果为 ()

```
void main()
{
    int x[]={1,2,3,4,5};
    int *p=x;
    printf("%d",*p++);
    printf("%d",*++p);
    printf("%d",*(++p));
    printf("%d",*(--p));
}
```

A. 1243

B. 1342

C. 1343

D. 1433

【提示】指针 p 指向数组 x 的首地址，即指向第 1 个元素，在本题中应注意*与++运算符的优先关系。

13. 以下程序运行结果为 ()。

```
void main()
{
    char s[]="HELLO";
    char *p=s;
    while(*p)
        printf("%c",*p+++32);
}
```

A. HELLO

B. hello

C. he

D. 程序运行会出错

【提示】本题将字符数组 s 中内容都加上了 32，即转化为了小写字母。

14. 以下程序运行结果为 ()。

```
#include <string.h>
void main()
{
    char s[]="avvsds";
    int a=strlen(s);
}
```



```
printf("%d",a);  
}
```

A. 5

B. 6

C. 7

D. 8

【提示】本题通过 `strlen()` 计算数组 `s` 的长度，输出结果至屏幕。

15. 以下程序运行结果为 ()。

```
void main()  
{  
    char s[]="abcdefgh";  
    char *p=s;  
    while(*p!= '\0')  
    {  
        printf("%c",*p);  
        p=p+2;  
    }  
}
```

A. aceg

B. abcd

C. acegh

D. bdfh

【提示】本题通过指针 `p` 进行输出操作，输入一个字符执行一次 `p=p+2` 直到字符串末尾为止。

二、填空题

1. 以下程序的运行结果为_____。

```
void main()  
{  
    int x[]={1,3,5,7,9},*p;  
    p=x;  
    printf("%d",*p+2);  
}
```

【提示】上述程序使指针 `p` 指向数组 `x` 首地址，输出 `*p+2` 的值。

2. 以下程序运行结果为_____。

```
void f(char *a,char b)  
{  
    char *c;  
    *c=*c+1;  
    b=b+1;  
}  
void main()  
{  
    char x='a',y='b';  
    f(&x,y);  
    printf("%c,%c",x,y);  
}
```



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】上述程序将 x 的指针及变量 y 传给函数，因此变量 x 的值会改变，而变量 y 的值不会改变。

3. 现有以下程序：

```
void main()
{
    char s1[]="Hello",s2[10];
    char *p1,*p2;
    p1=s1;
    p2=s2;
    scanf("%s",p2);
    printf("%s,%s",p1,p2);
}
```

若从键盘输入字符串 happy，则输出结果为_____。

【提示】本题通过指针 p1 和 p2 输出字符串。

4. 现有以下程序：

```
#include <stdio.h>
#include <string.h>
int s(char *p1,char *p2)
{
    while(*p1)
        p1++;
    while(*p2)
        *p1=*p2++;
    *p1='\0';
    return p1;
}
void main()
{
    char s1[50],s2[50];
    char *p1,*p2;
    p1=s1;
    p2=s2;
    gets(p1);
    gets(p2);
    printf("%s",s(p1,p2));
}
```

若输出字符串 abcd 和 shkg，则其运行结果为_____。

【提示】本题 s()函数将字符串 s2 连接至字符串 s1 后，然后在主函数 main()中输出字符串 s1。

5. 以下程序的功能是从键盘输入一个字符串，输出该字符串至屏幕，补全下面的空白处。

```
void main()
{
```



```
char s[100],*p;
p=____;
scanf("____",p);
while(*p!= '\0')
    putchar(____);
}
```

【提示】上述程序通过指针 *p* 指向数组 *s*，输出字符串至屏幕。

6. 以下程序的运行结果为_____。

```
void main()
{
    int i;
    int x[6]={1,2,3,4,5,6};
    int *n[6];
    int **p;
    for(i=0;i<6;i++)
        n[i]=x+i;
    p=n;
    for(i=0;i<6;i++,p++)
        printf("%d ",**p);
}
```

【提示】上述程序利用二级指针 *p* 输出数组 *x* 中的元素。

7. 以下程序运行结果为_____。

```
void main()
{
    int a[]={1,2,3,4,5,6};
    int *p=a+2;
    printf("%d",*++p);
}
```

【提示】执行 **p=a+2* 后指针 *p* 指向数组的第 3 个元素，因此 **++p* 指向数组的第 4 个元素。

8. 以下程序运行结果为_____。

```
void main()
{
    char s[]="Hello world! ";
    char *p=s;
    while(*p!= 'l'&&*p!='\0')
    {
        printf("%c",*p);
        p++;
    }
}
```

【提示】本题通过指针 *p* 输出字符数组 *s* 中的内容，若指针 *p* 中的内容不等于字符 *l* 并且未



移动到末尾，则输出字符至屏幕。

9. 以下程序运行结果为_____。

```
void main()
{
    int a[8],i,j;
    int *p;
    for(i=0;i<8;i++)
    {
        j=i;
        a[i]=3*j+1;
    }
    for(p=a;p<a+8;p++)
    printf("%d ",*p);
}
```

【提示】上述程序通过指针 p 输出数组 a 中的每个元素。

10. 以下程序运行结果为_____。

```
void main()
{
    int a[]={1,3,5,7,9};
    int *p=a+3;
    printf("%d",*p--);
}
```

【提示】执行 *p=a+3 后指针 p 指向数组第 4 个元素，因此 *p-- 指向数组的第 4 个元素。

11. 以下程序运行结果为_____。

```
void main()
{
    int a[]={1,2,3,4,5,6,7,8,9,10},*p;
    p=a+6;
    printf("%d,%d,%d",*(a+5),*p++,*++p);
}
```

【提示】p=a+6 后指针 p 指向数组 a 的第 7 个元素，因此 *p++ 和 *++p 分别指向数组的第 7 个元素和第 8 个元素，但其中还要注意 p 值的变化。

12. 以下程序运行结果为_____。

```
void main()
{
    int a[5]={1,45,12,3,41};
    int *p=a,i,m;
    for(i=0;i<4;i++)
    if(a[i]<a[i+1])
    m=a[i+1];
}
```



```
printf("%d",m);  
}
```

【提示】本题通过 for 循环将数组 a 中最大的元素保存至变量 m 中，最后进行输出。

13. 以下程序运行结果为_____。

```
void main()  
{  
    char s1[]="C Language",s2[]="Program";  
    char *p1,*p2;  
    p1=s1;  
    p2=s2;  
    while(*p1!= '\0')  
        p1++;  
    *p1+=*p2++;  
    *p1='\0';  
    printf("%s",p1);  
}
```

【提示】本题通过指针将字符数组 s2 中的内容连接至字符数组 s1 后，再输出结果。

14. 以下程序运行结果为_____。

```
void f(int a,int b)  
{  
    int t;  
    t=a;  
    a=b;  
    b=t;  
}  
void main()  
{  
    int x=5,y=6;  
    f(x,y);  
    printf("%d,%d",x,y);  
}
```

【提示】本题将 x 和 y 作为实参传递给函数交换值大小，但由于没有对地址进行操作，因此主函数中 x 和 y 的值仍然不会改变。

三、编程题

1. 编写一个函数，实现字符串的复制功能。

【提示】设有两个字符串 s1 和 s2，要将字符串 s2 复制给字符串 s1 可以设置两个指针 p1 和 p2。p1 指向字符串 s1 的首地址，p2 指向字符串 s2 的首地址，p1 和 p2 同时移动并将 p2 中的内容赋给 p1 即可实现字符串的复制功能。

【核心代码】

```
void s(char *p1,char *p2)  
{
```



```
while(*p2)
{
    *p1=*p2;
    p1++;
    p2++;
}
```

2. 从键盘输入 5 个字符串，对这 5 个字符串从小到大排列并输出排列结果。

【提示】从键盘输入 5 个字符串可以利用 `gets()` 函数或 `scanf()` 函数来实现。对 5 个字符串排序则可以用冒泡排序或选择排序法来进行排序，最后利用 `printf()` 或 `puts()` 函数输出排好序的结果。

【核心代码】

```
char *s[5];
for(i=0;i<5;i++)
    scanf("%s",s[i]);
for(i=0;i<4;i++)
    for(j=i+1;j<5;j++)
        if(strcmp(s[i],s[j])>0)
        {
            t=s[i];
            s[i]=s[j];
            s[j]=t;
        }
for(i=0;i<5;i++)
    printf("%s\n",s[i]);
```

3. 编写一个函数，通过指针实现两个数的交换。

【提示】通过函数来交换两个数，则需对这两个数的地址进行操作，交换两个地址中的内容。可以定义两个指针，分别指向这两个数，再将这两个指针作为参数传递至函数中，交换指针中的内容即可实现交换。

【核心代码】

```
char *p,*q;
p=&x;
q=&y;
void change(char *p,char *q)
{
    t=*p;
    *p=*q;
    *q=t;
}
```

4. 从键盘输入一个字符串，分别统计字符串中字母、数字及其他字符的个数并输出。

【提示】统计一个字符串中的字母、数字及其他字符的个数，可以通过计数器来计算。设



置三个计数器 a、b、c，初始值都为 0，再对字符串中的字符进行判断。若为字母则 a 加 1，若为数字则数字加 1，若为其他字符则 c 加 1。最后输出 a、b、c 的数值即为字母、数字及其他字符的个数。

【核心代码】

```
int a=0,b=0,c=0;
while(*p!= '\0')
{
    if((*p>='a'&&*p<='z')||(*p>='A'&&*p<='Z'))
        a++;
    else if(*p>='0'&&*p<='9')
        b++;
    else
        c++;
}
printf("%d,%d,%d\n",a,b,c);
```

5. 从键盘输入一个字符串和一个数字 n，要求从字符串第 n 个字符开始至字符串的末尾重新组成一个字符串输出。

【提示】将字符串第 n 个位置至末尾重新组成一个字符串，可以通过指针来实现。指针指向字符串的起始位置，从键盘输入 n，再将指针移动至字符串的第 n 个位置。定义另外一个指针，两个指针同时移动，将剩余的字符串通过指针进行复制，最后输出指针中的字符串。

【核心代码】

```
scanf("%s",p);
scanf("%d",&n);
for(i=0;i<n-1;i++)
    p++;
while(*p!= '\0')
    q++=p++;
printf("%s",q);
```

6. 编写一个程序，要求利用指针求任意数据的绝对值。

【提示】求数据的绝对值，可以另外编写一个函数，通过指针作为参数来传递，并将指针的地址返回至主函数输出结果。

【核心代码】

```
void f(int *p)
{
    if(*p<0)
        *p=-*p;
    return *p;
}
void main()
{
    s=f(&x);
```



```
printf("%d",s);  
}
```

7. 从键盘输入 10 个数至数组中，然后将其中的元素进行逆序存放后输出。

【提示】定义整型数组保存这 10 个数，通过指针将数组中的元素逆序存放，再输出其元素至屏幕。

【核心代码】

```
for(i=0;i<10;i++)  
scanf("%d",&x[i]);  
p=x;  
for(i=0;i<=5;i++)  
{  
    t=*(p+i);  
    *(p+i)=*(p+9-i);  
    *(p+9-i)=t;  
}
```

8. 从键盘输入一个字符串，将其逆序输出。

【提示】定义字符数组保存字符串，要将其中的字符串逆序输出，可以先将指针移动到字符串的末尾，然后向前移动并输出。

【核心代码】

```
scanf("%s",s);  
p=s;  
while(*p!= '\0')  
p++;  
p--;  
while(1)  
{  
    printf("%c",*p);  
    if(p==s)  
        break;  
}
```

9. 从键盘输入两个字符串 s1 和 s2，将 s2 连接至 s1 后输出。

【提示】将字符串 s2 连接到 s1 后面，可以通过字符指针来实现。指针 p1 指向 s1 的末尾，指针 p2 指向 s2 的开头，然后通过指针 p1 和 p2 将字符串 s2 连接到字符串 s1 之后，最后在字符串 s1 后加上\0 结束符即可。

【核心代码】

```
gets(s1);  
gets(s2);  
while(*p1!= '\0')  
p1++;  
while(*p2!= '\0')  
*p1++=*p2++;
```



```
*p1='\0';
```

10. 从键盘输入两个字符串，编写一个程序判断两个字符串是否相等。

【提示】比较两个字符串的大小，通过指针来实现。指针 p1 和 p2 分别指向两个字符串，通过 p1 和 p2 逐个字符进行比较再输出。

【核心代码】

```
gets(s1);  
gets(s2);  
c=strcmp(s1,s2);
```

第 9 章 函 数

函数是 C 语言程序的基本组成单位，一个程序由一个或多个函数组成。函数是一个独立的模块，利用函数可以实现程序模块化，使程序简洁清晰。C 语言中的函数包括库函数和自定义函数，库函数是指包含在头文件中的系统已定义好的函数，自定义函数为用户自己定义的函数。

用户自定义函数可以直接与 `main()` 函数放在同一个文件中，也可以单独放在其他的文件中，分别进行编译，最后链接在一起，形成可执行文件*.exe。函数在 C 语言中起着至关重要的作用，因此读者应认真学习本章。

本章主要涉及的知识点有：

- 函数的声明；
- 函数应用；
- 递归函数；
- 函数参数；
- 内部变量和外部变量。



9.1 函数定义和调用

在 C 语言中，可以将一段代码封装起来，在需要时直接调用，这段代码称为函数。在本节中，将讲解函数的定义及其使用方法。

9.1.1 定义函数

根据函数有无参数，可以将函数分为有参函数和无参函数两种。

无参函数的定义形式如下：

```
类型 函数名(void)
{
    可执行语句
}
```

有参函数定义形式如下：

```
类型 函数名(类型 形参 1, 类型 形参 2, ... )
{
    可执行语句
}
```

函数和变量一样，也需先定义才能使用。若在文件的开头定义了函数，则不需加上函数声明，若定义函数在调用之后，则必须要加上声明，否则会出错。其声明形式如下：

```
类型 函数名(类型 形参 1，类型 形参 2，...)
或 类型 函数名()
```



注意：

- (1) 当函数只需实现特定的功能而不需返回值时，可以定义为 void 类型，若函数需返回值时，则应定义为其类型。
- (2) 形参若多于一个，则用逗号分开。
- (3) 函数体内部定义的变量为局部变量，只在函数体内有效。同一个局部变量可以在多个函数中出现，不会出错。
- (4) 一个函数可以调用另一个函数，但不能在函数体内定义另一个函数。
- (5) 当一个函数需返回值时，可以利用 return 语句来实现。

【范例 9.1】 定义一个函数，从键盘输入两个数，求两个数中的大者。

分析：定义一个函数，有两个参数，返回值为较大者。在主函数中输入两个数据，调用函数并将这两个数据作为实参传递给函数，调用 printf()函数输出运算得出的结果。

范例 9.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int max(int n,int m);                            /*自定义函数 max( )原型声明*/
05      int n1,n2;
06      printf("输入两个数: ");
07      scanf("%d%d",&n1,&n2);
08      printf("the max is %d\n",max(n1,n2));            /*调用 max( )函数输出两者中的大者*/
09      getch();
10  }
11  int max(int n,int m)                                /*自定义函数 max( )*/
12  {
13      if(n>m)                                           /*若 n 大于 m 则返回值 n*/
14          return n;
15      else                                             /*否则返回值 m*/
16          return m;
17  }
```

【代码分析】 本例为函数的简单范例，详细代码分析如下：

- 第 4 行为 max()函数的声明。因为函数代码在调用函数之后，因此必须加上函数的声明。
- 第 8 行，调用 max()函数并将 n1 和 n2 作为实参传递给函数。
- 第 11~17 行，自定义 max()函数。比较 n 和 m 两个数的大小，返回两个数中的较大者。

【运行结果】 该程序的执行结果如图 9-1 所示。

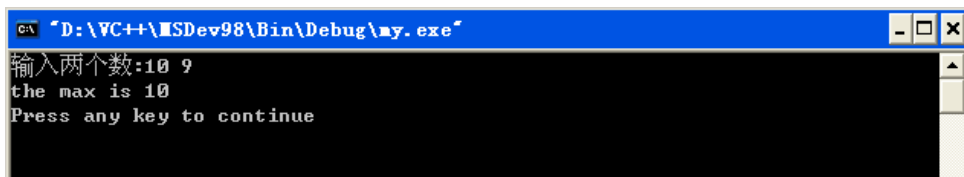


图 9-1 范例 9.1 结果图

9.1.2 调用函数

调用函数的形式如下：

函数名（实参表）；

其中实参是指确定的变量或表达式，各个实参用逗号分开。实参的个数应与形参的个数一致，与形参是一一对应的关系。若无参数，括号也不能省去。

函数调用方式有以下 4 种。

1. 函数语句调用方式

这种方式是指调用函数作为一个独立的语句放在主函数 `main()` 中，其中函数没有返回值。其示例如下：

```
void f()  
{  
    int a=1;  
    a++;  
    printf("%d",a);  
}  
void main()  
{  
    f();  
}
```

上述程序在主函数 `main()` 中调用 `f()` 函数作为一条独立的语句。其中 `f()` 函数为 `void` 类型，没有返回值，只是完成 `a` 加 1 的操作并将其输出。

2. 函数表达式调用方式

这种方式是将函数用于表达式的计算，其中函数都有一个确定的返回值，用来参与表达式的计算。其示例如下：

```
int max(int m,int n)  
{  
    int s;  
    s=(m>n)?m:n;  
    return(s);  
}  
void main()  
{
```



```
int x=20,y=30,z;  
z=10*max(x,y);  
printf("z=%d",z);  
}
```

上述程序将 `max()` 函数应用于表达式 `10*max(x,y)` 中，其中 `max()` 函数返回 `x` 和 `y` 中的较大者，再乘以 10 计算得出结果。

3. 函数参数调用方式

这种方式是将函数作为另一个函数的参数，其中函数必须有一个返回的值，用来作为函数的参数。其示例如下：

```
int f(int n)  
{  
    int y=1;  
    while(n>1)  
        y=y*n;  
    return(y);  
}  
void main()  
{  
    int x;  
    scanf("%d",&x);  
    printf("%d!=%d",x,f(x));  
}
```

上述程序中将 `f()` 函数用于 `printf()` 函数中作为参数，用于计算 `x!` 并输出结果。

4. 库函数调用方式

这种方式是直接调用 C 语言中的库函数。但在调用库函数之前，应记住包含相应的头文件，其示例如下：

```
#include <math.h>  
void main()  
{  
    int x,y;  
    scanf("%d",&x);  
    y=fabs(x);  
    printf("%d",y);  
}
```

上述程序中调用数学库中的函数 `fabs()`，其功能为返回一个数的绝对值，在引用该函数之前必须要先包含 `math.h` 头文件。

9.1.3 函数的返回值

函数的返回值可以为某一类型的值，也可以不返回任何值。



1. 返回某种类型的值

函数返回某一类型的值可以通过 `return` 来实现。根据具体的情况设置一个或多个 `return` 语句，`return` 语句可以返回计算结果并使程序返回至调用函数的语句处继续执行程序。

`return` 语句的形式如下：

```
return(表达式);
```

其中表达式可以为符合 C 语言的语法的任意表达式，其值类型应与函数类型一致。

【范例 9.2】 计算函数 $y = \begin{cases} 2x-1 & (x > 3) \\ x^2+1 & (x \leq 3) \end{cases}$

分析：上述函数为分段函数， x 的值不同则函数计算得出的结果也不同。因此可对 x 的值进行判断，若 x 的值大于 3，则返回 $2x-1$ 的值，否则返回 x^2+1 的值。

范例 9.2 代码实现

```
01  #include <stdio.h>
02  int f(int x)
03  {
04      if(x>3)                                /*若 x 大于 3 则输出 x 的平方加 1*/
05          return(x*x+1);
06      else                                    /*否则输出 2x-1*/
07          return(2*x-1);
08  }
09  void main()
10  {
11      int a;
12      scanf("%d",&a);
13      printf("y=%d\n",f(a));
14  }
```

【代码分析】 本例中函数根据条件不同返回不同的值，详细代码分析如下：

第 2~8 行为自定义函数 `f()`。该函数 x 的值不同则返回不同的值，若 x 的值大于 3，则返回 $2x-1$ 的值，否则返回 x^2+1 的值。

【运行结果】 该程序的执行结果如图 9-2 所示。

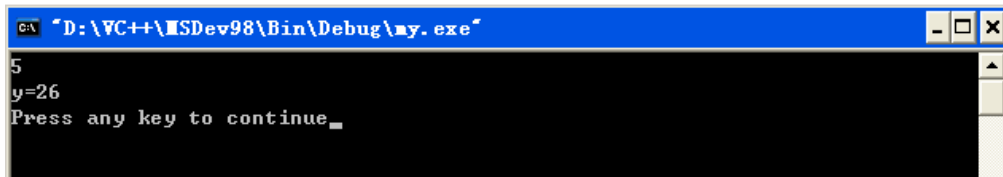


图 9-2 范例 9.2 结果图

2. 不返回任何值

函数若定义为 `void` 类型，则其不返回任何值，函数中没有 `return` 语句。函数执行完后即会自动返回至调用函数的语句处继续执行程序。



9.2 变量的生存期和作用域

在 C 语言中，每一个变量都有其生存期和作用域。变量声明的位置决定了其生存期和作用域，变量可以分为函数内部变量和函数外部变量。

9.2.1 函数内部变量

函数内部变量是指在函数体内声明的变量。它只在函数体内部有效，即其作用域为函数体内，在函数体外变量无法使用。例如：

```
int f(int x)
{
    int y,z;
}
void main()
{
    int a,b;
}
```

上述程序中，y，z，a，b 都为函数内部变量，其中 y，z 的作用域为 f()函数内，a，b 的作用域为 main()函数体内。



注意：

- (1) 主函数 main()中的变量也为局部变量，在其他函数中不能使用。
- (2) 不同函数可以定义相同的变量名，其作用域只在本身的函数体内有效，不会起冲突。
- (3) 复合语句中定义的内部变量，只在复合语句内有效。

【范例 9.3】通过下面的例子，理解函数内部变量。

分析：局部变量只在函数体内有效，在函数体外不起作用。

范例 9.3 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      void f();
05      int x=5;
06      f();                                /*调用自定义函数 f()*/
07      printf("x int main is %d\n",x);
08  }
09  void f()                                /*自定义函数 f()*/
10  {
```



```
11    int x=8;
12    printf("x in f() is %d\n",x);
13 }
```

【代码分析】本例为函数内部变量的简单范例，详细代码分析如下：

- 第 6 行，调用 f() 函数，输出函数内部变量 x 的值。

【运行结果】该程序的执行结果如图 9-3 所示。

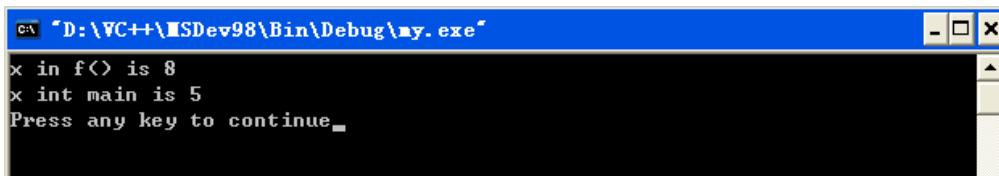


图 9-3 范例 9.3 结果图

9.2.2 函数外部变量

函数外部变量是指在函数外部定义的变量，它不属于任何一个函数，其作用域为从其定义位置至程序的末尾，可以被其他任意函数使用，也称为全局变量。例如：

```
int a,b;
void f1()
{
    int x;
}
void f2()
{
    int y;
}
```

a, b 为全局变量，可以被 f1() 和 f2() 函数使用。

若在同一程序中，全局变量与局部变量同名，则局部变量在其作用域内会取代全局变量。

例如：

```
int x=1,y=3;
void main()
{
    int x=2;
    printf("%d,%d",x,y);
}
```

上述程序中开头 x, y 被定义为全局变量，在 main() 函数中 x 被定义为局部变量，因此会掩盖全局变量 x，即输出的结果为 2,3。

【范例 9.4】从键盘输入 5 个学生的成绩，编写一个函数计算这 5 名学生成绩的最高分和平均分。



分析：定义一个一维数组保存 5 名学生的成绩，编写一个函数计算最高分和平均分，最后可以通过函数的返回值返回结果，也可以直接对全局变量进行操作。

范例 9.4 代码实现

```
01  #include <stdio.h>
02  float max;                      /*定义全局变量max*/
03  float f(float a[],int n)
04  { int i;
05    float x,sum=0;
06    max=a[0];                      /*将a[0]赋值给变量max*/
07    for(i=0;i<n;i++)
08    {
09      if(a[i]>max)                  /*若a[i]大于max 则将a[i]赋值给max*/
10        max=a[i];
11      sum=sum+a[i];                /*计算各个元素的累加和*/
12    }
13    x=sum/n;                       /*求其平均分*/
14    return(x);
15  }
16  void main()
17  {
18    float x,s[5];
19    int i;
20    for(i=0;i<5;i++)
21      scanf("%f",&s[i]);
22    x=f(s,5);                      /*调用自定义函数f()*/
23    printf("the average is %f \n",x);
24    printf("the max is %f\n",max);
25  }
```

【代码分析】本例为函数外部变量的简单范例，详细代码分析如下：

- 第 2 行，定义了一个全局变量 `max`，用于保存 5 名学生成绩中的最大数。
- 第 3~15 行为自定义函数。此函数的功能为计算 5 名学生的平均分和最高分。
- 第 22 行，调用 `f()` 函数计算其平均分和最高分，其中数组 `s` 和学生个数 5 作为实参传递给函数 `f()`。

【运行结果】该程序的执行结果如图 9-4 所示。

```
C:\ "D:\VC++\MSDev98\Bin\Debug\my.exe"
78 67 88 79 84
the average is 79.200000
the max is 88.000000
Press any key to continue.
```

图 9-4 范例 9.4 结果图

【范例 9.5】输入圆形的半径，求圆形的面积和周长，要求利用全局变量进行计算。



由浅入深学 C 语言——基础、进阶与必做 430 题

分析：全局变量可以被程序中的任意函数调用，编写一个函数对全局变量进行操作计算圆形的面积和周长，最后输出计算的结果即可。

范例 9.5 代码实现

```
01  #include <stdio.h>
02  float s,z;                                /*定义全局变量 s 和 z*/
03  void f(float r)
04  {
05      s=3.14*r*r;
06      z=2*3.14*r;
07  }
08  void main()
09  {
10      float r;
11      printf("输入圆的半径: ");
12      scanf("%f",&r);
13      f(r);
14      printf("圆的面积为%f\n",s);           /*输出圆的面积*/
15      printf("圆的周长为%f\n",z);
16  }
```

【代码分析】本例为全局变量和函数的应用范例，详细代码分析如下：

- 第 2 行，定义了两个全局变量 s 和 z。其中 s 用来保存圆的面积，z 用来保存圆的周长。
- 第 3~7 行为自定义 f() 函数。此函数的功能为计算圆的面积和周长，分别保存至变量 s 和 z 中，其中 r 作为函数的参数。
- 第 13 行，调用函数 f()，其中 r 作为实参传递给函数。

【运行结果】该程序的执行结果如图 9-5 所示。

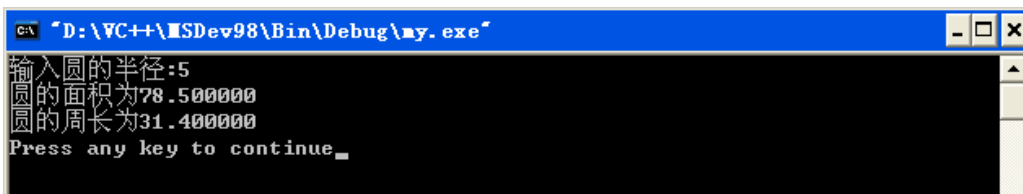


图 9-5 范例 9.5 结果图



注意：

- (1) 全局变量可以被多个函数使用，使用起来很方便。但由于全局变量可被多个函数使用，其值很容易改变。当有很多函数使用全局变量时，程序就很容易出错。
- (2) 在程序中，若局部变量与全局变量同名，则不会引起冲突，局部变量在其作用域内会掩盖全局变量。
- (3) 全局变量的作用域为其定义位置到程序的末尾，其生存期为程序的开始至结束。



9.3 函数的实参和形参

函数的参数分为实参和形参。在函数调用的过程中，实参的值将会传递给函数的形参，相反形参不能传递给实参。函数参数的传递分为传值方式和传址方式两种，在本节中将讲解这两种方式的概念及其区别。

9.3.1 传值方式

当要将常量、变量传递给函数时，即可采用传值方式。传值方式形参值的改变不会影响实参的值，其中形参的类型和个数应与实参一致。

【范例 9.6】编写一个函数，通过传值方式引用函数。

分析：传值方式是直接将实参（变量）传递给函数中的形参（另一个变量），然后通过函数中的变量进行操作计算得出结果。

范例 9.6 代码实现

```
01  #include <stdio.h>
02  void f(int m,int n)
03  {
04      int t;                                /*通过变量 t 交换 m 和 n 的值*/
05      t=m;
06      m=n;
07      n=t;
08      printf("m=%d,n=%d\n",m,n);          /*输出交换后 m 和 n 的值*/
09  }
10  void main()
11  {
12      int m=30,n=40;
13      f(m,n);
14      printf("m=%d,n=%d\n",m,n);
15  }
```

【代码分析】本例通过传值方式引用函数，详细代码分析如下：

- 第 2~9 行为自定义函数 f()。此函数的功能为交换两个数的数值并输出交换后的结果，其中 m 和 n 作为函数的形参用来接收实参的值。
- 第 13 行，调用 f() 函数，其中 m 和 n 作为实参传递给函数。

【运行结果】该程序的执行结果如图 9-6 所示。



注意：

(1) 传值方式是将实参的值传递给形参。形参可以与实参同名，形参值的变化不会影响实参值的变化。

(2) 传值方式中 return 语句只能返回一个值。

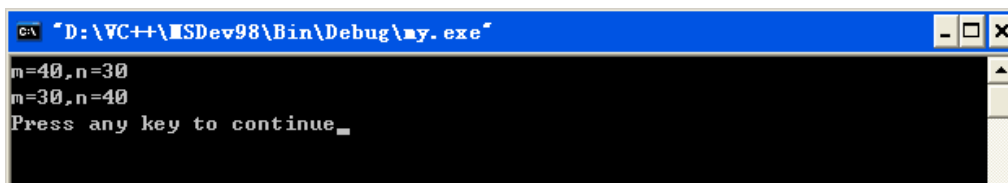


图 9-6 范例 9.6 结果图

9.3.2 传址方式

传址方式是将实参的地址传递给形参。函数将会对这个地址进行操作，因此形参值的变化会影响实参的值。在传址方式中，实参可以为变量地址或指针名、一维数组、字符数组或多维数组等。

1. 变量地址作实参

【范例 9.7】 编写一个函数，通过传址方式交换两个数的数值。

分析：传址方式是指将变量的地址传递给形参，对变量的地址进行操作。因此传址方式中形参值的变化会影响实参的值。

范例 9.7 代码实现

```
01  #include <stdio.h>
02  void f(int *m,int *n)
03  {
04      int t;                                /*通过变量 t 交换*m 和*n 的值*/
05      t=*m;
06      *m=*n;
07      *n=t;
08      printf("m=%d,n=%d\n",*m,*n);        /*输出交换后*m 和*n 的值*/
09  }
10  void main()
11  {
12      int m=30,n=40;
13      f(&m,&n);
14      printf("m=%d,n=%d\n",m,n);
15  }
```

【代码分析】 本例通过传址方式引用函数，详细代码分析如下：

- 第 2~9 行为自定义函数 f()。此函数的功能为交换传入函数的两个数，其中指针 m 和 n 作为函数的形参用来接收实参的地址。
- 第 13 行，调用 f() 函数，其中将变量 m 和 n 的地址作为实参传递给函数。

【运行结果】 该程序的执行结果如图 9-7 所示。

2. 一维数组名作实参

一维数组名表示数组的首地址，因此可以作为实参传递给函数。

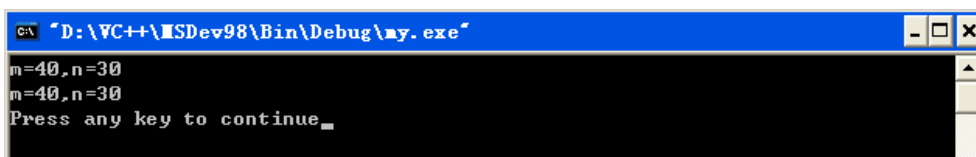


图 9-7 范例 9.7 结果图

【范例 9.8】从键盘输入两个字符串，比较这两个字符串是否相等。若两个字符串相等则输出“相等”，否则输出“不相等”。

分析：用两个字符数组来保存输入的字符串，对两个数组逐个进行比较，若相等则输出“相等”信息，否则输出“不相等”信息。

范例 9.8 代码实现

```
01  #include <stdio.h>
02  void f(char s1[],char s2[])
03  {
04      int i=0;
05      while(s1[i]==s2[i]&& s1[i]!='\0')    /*while 循环比较两个字符串是否相等*/
06          i++;
07      if(s1[i]=='\0'&& s2[i]=='\0')        /*若移动到字符串末尾*/
08          printf("相等\n");                /*输出相等信息*/
09      else                                  /*否则输出不相等信息*/
10          printf("不相等\n");
11  }
12  void main()
13  {
14      char s1[30],s2[30];
15      printf("输入两个字符串\n");
16      gets(s1);                            /*从键盘获取字符串保存至数组 s1 中*/
17      gets(s2);
18      f(s1,s2);
19  }
```

【代码分析】本例将数组名作为实参引用函数，详细代码分析如下：

- 第 2~11 行为自定义函数 f()。该函数通过逐个字符的比较来比较两个字符串是否相等，若相等则输出“相等”，否则输出“不相等”。
- 第 16、17 行，通过 gets() 函数从键盘获取字符串。

【运行结果】该程序的执行结果如图 9-8 所示。

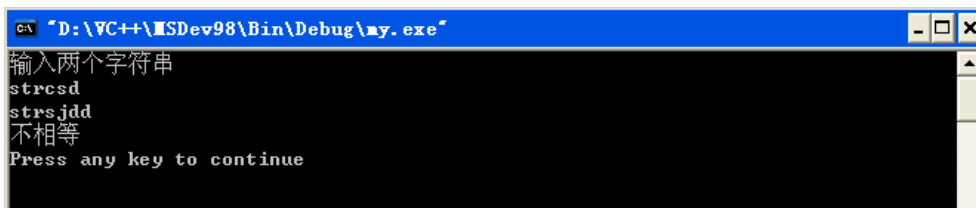


图 9-8 范例 9.8 结果图



3. 字符串作实参

字符串作为实参，形参应为字符型的指针变量。

【范例 9.9】编写一个程序，计算一个字符串中 s 字符出现的次数。

分析：计算一个字符串中 s 字符出现的次数，可以设置一个计数器。计数器初始值为 0，将字符 s 与字符串中的字符逐个进行比较，若相等则计数器加 1，否则计数器的数值不变。

范例 9.9 代码实现

```
01  #include <stdio.h>
02  void f(char *s)
03  {
04      char c='s';
05      int i=0,m=0;
06      while(s[i]!='\0')           /*while 循环判断是否到字符串末尾*/
07      {
08          if(s[i]=='s')           /*若字符等于 s 则计数器加 1*/
09              m++;
10              i++;
11      }
12      printf("the num is %d\n",m);
13  }
14  void main()
15  {
16      char s[50];
17      gets(s);                     /*从键盘获取字符串至数组 s 中*/
18      f(s);
19  }
```

【代码分析】本例将字符串作为实参引用函数，详细代码分析如下：

- 第 3~13 行为自定义函数 f()，此函数的功能为判断一个字符串中字符 s 的个数。

【运行结果】该程序的执行结果如图 9-9 所示。

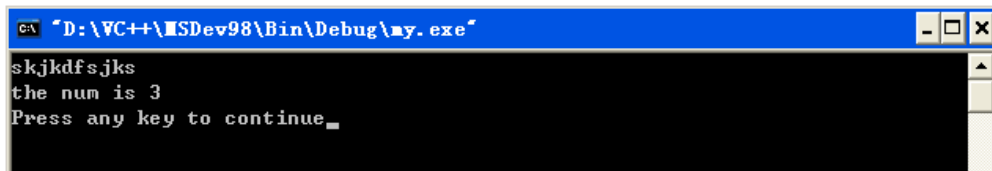


图 9-9 范例 9.9 结果图



9.4 函数的嵌套和递归

在 C 语言中，在一个函数中允许调用另外的函数或本身函数。一个函数调用其他函数称为函数的嵌套调用，若调用本身的函数称为函数的递归调用。在本节中将讲解函数的嵌套和递归



概念及其应用。

9.4.1 函数的嵌套

函数的嵌套是指在一个函数中又调用另外的函数，下面通过一个例子来了解一下函数的嵌套调用。

【范例 9.10】从键盘输入一个数 x ，计算 $1^5+2^5+3^5+\cdots+x^5$ 的值。

分析：本例可以采用函数嵌套的方法，将计算 x 的 5 次方定义为函数 $f()$ ，计算 1 到 x 的 5 次方之和定义为函数 $s()$ 。主函数输入数字 x ，在函数 $s()$ 中调用 $f()$ 函数计算上述表达式的结果输出即可。

范例 9.10 代码实现

```
01  #include <stdio.h>
02  long f(int x)
03  {
04      long p=x;int i;                /*定义一个长整型变量p和整型变量i*/
05      for(i=1;i<5;i++)                /*通过for循环计算p的5次方*/
06          p=p*x;
07      return(p);                    /*返回p值*/
08  }
09  long s(int x)
10  {
11      long s=0;
12      int i;
13      for(i=0;i<x;i++)                /*通过for循环计算0到x的5次方累加和*/
14          s=s+f(i);
15      return(s);
16  }
17  void main()
18  { int x;
19      printf("输入数字 x: ");
20      scanf("%d",&x);
21      printf("the sum is %d\n",s(x));
22  }
```

【代码分析】本例为函数的嵌套范例，详细代码分析如下：

- 第 2~8 行为自定义函数 $f()$ 。该函数的功能为用来计算一个数 x 的 5 次方的值并将结果返回。
- 第 9~16 行为自定义函数 $s()$ 。该函数的功能为计算 1 到 x 的所有数 5 次方之和，其中使用了函数 $f()$ 用来计算每一个数的 5 次方值。 s 定义为长整型，因为结果可能超过整型数据的最大上限。
- 第 21 行，调用函数 $s()$ 计算并输出结果。

【运行结果】该程序的执行结果如图 9-10 所示。

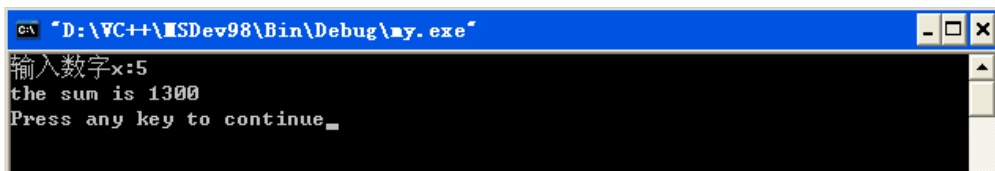


图 9-10 范例 9.10 结果图

9.4.2 函数的递归

函数的递归是指在一个函数中调用其本身函数，递归调用是函数嵌套的一种特殊情况。C 语言中函数递归应有其终止条件，即满足该条件则停止递归调用，逐层返回。

【范例 9.11】 现有 6 个人，问第 6 个人多少岁？第 6 个人说他比第 5 个人大 3 岁，第 5 个人说他比第 4 个人大 3 岁，依次问下去，最后一个人说他 13 岁。求第 6 个人多大岁数。

分析：设 6 个人的年龄分别为 $a_1 \sim a_6$ ，则有如下关系：

```
a1=13;
a2=a1+3;
a3=a2+3;
a4=a3+3;
a5=a4+3;
a6=a5+3;
```

因此可以用递归函数来求第 6 个人的岁数。

范例 9.11 代码实现

```
01  #include <stdio.h>
02  int a(int n)
03  {
04      int x;
05      if(n==1)                /*若 n 等于 1 则 x 值为 13*/
06          x=13;
07      else                    /*否则将 a(n-1)+3 值赋给变量 x*/
08          x=a(n-1)+3;
09      return x;               /*返回 x 值*/
10  }
11  void main()
12  {
13      printf("result is %d\n",a(6)); /*调用 a()函数计算输出结果*/
14  }
```

【代码分析】 本例为递归函数应用范例，详细代码分析如下：

- 第 2~10 行为用户自定义函数 `a()`。此函数为递归函数，在函数体内部又调用函数本身，若 n 等于 1 则停止递归调用，逐层返回计算。
- 第 13 行，调用 `a()` 函数计算第 6 个人的岁数并将结果输出。

【运行结果】该程序的执行结果如图 9-11 所示。

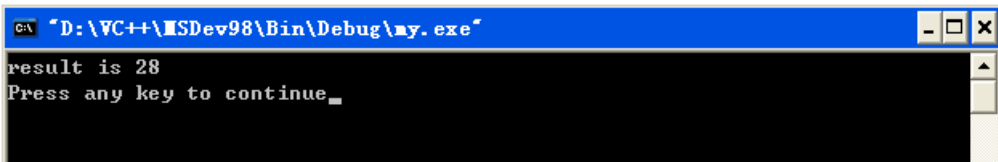



图 9-11 范例 9.11 结果图



9.5 函数应用举例

【范例 9.12】编写一个函数，计算两个数的差值并将结果输出。

分析：编写函数计算两个数的差值，可以将这两个数作为参数传递给函数，计算两个数的差值。最后在主函数中调用函数，输出计算结果。

范例 9.12 代码实现

```
01  #include <stdio.h>
02  int f(int a,int b)
03  {
04      int c;
05      c=a-b;                      /*将 a-b 的值赋值给变量 c*/
06      return(c);                 /*返回 c 值*/
07  }
08  void main()
09  {
10      int x,y,z;
11      printf("输入两个数: ");
12      scanf("%d%d",&x,&y);
13      z=f(x,y);                  /*调用函数 f()求差值*/
14      printf("两个数的差值为%d\n",z);
15  }
```

【代码分析】本例为函数应用范例，详细代码分析如下：

- 第 2~7 行，用户自定义函数 f()。此函数的功能为计算 a 和 b 两个数的差值，并将计算结果返回。

【运行结果】该程序的执行结果如图 9-12 所示。

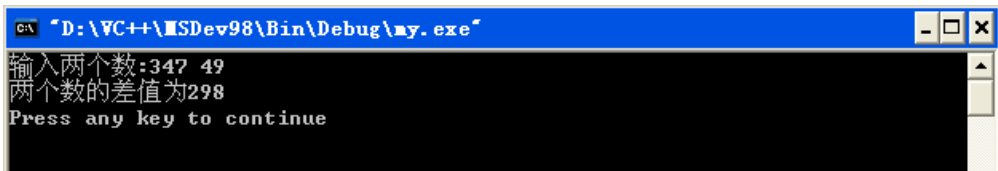


图 9-12 范例 9.12 结果图



【范例 9.13】从键盘输入一个十进制数，将其转换为二进制数输出结果至屏幕。

分析：将一个十进制数转换为二进制数，可以采用除以 2 取余数的方式，直到不能整除 2 为止。将十进制数除以 2 的余数保存至一个数组中，最后输出该数组即为十进制对应的二进制数。

范例 9.13 代码实现

```
01  #include <stdio.h>
02  void s(int x)
03  {
04      int i=0,j,s[100];
05      while(1)                                /*while 循环,条件值始终为 1*/
06      {
07          s[i++]=x%2;                          /*将 x 除以 2 的余数赋值给变量 s[i]*/
08          x=x/2;                                /*将 x 除以 2 结果赋值给变量 x*/
09          if(x==0)                            /*若 x 等于 0 则退出 while 循环*/
10              break;
11      }
12      for(j=i;j>=0;j--)                        /*倒着输出数组 s 中的元素*/
13          printf("%d",s[j]);
14      }
15  void main()
16  {
17      int a;
18      printf("输入要转换的十进制数: ");
19      scanf("%d",&a);
20      s(a);
21      printf("\n");
22  }
```

【代码分析】本例通过函数转化十进制数为二进制数，详细代码分析如下：

- 第 2~14 行，自定义函数 s()。此函数的功能为将十进制数转换为二进制数，它采用的是模 2 取余的方法，将余数保存至一个数组中，最后输出数组中的结果。
- 第 7 行，将 x 除以 2 的余数保存至 x 数组中。
- 第 8 行，将 x 整除 2 的结果重新赋给变量 x。
- 第 9、10 行，若 x 的值为 0，则表示数值转换完毕，跳出 while 循环。
- 第 12、13 行，倒序输出数组 s 中的元素，即从高位到低位依次输出。
- 第 20 行，调用函数 s()，其中变量 a 作为参数传递给函数。

【运行结果】该程序的执行结果如图 9-13 所示。

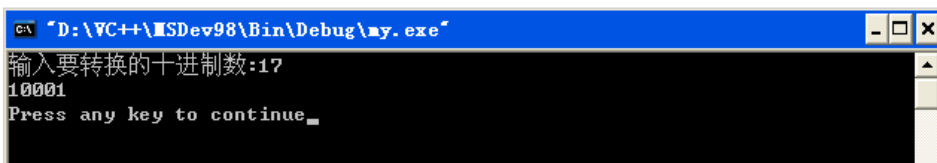


图 9-13 范例 9.13 结果图



9.6 小结

函数是 C 语言重中之重，用 C 语言编写的每一个程序中都必须有 `main()` 函数。通过函数可以实现程序的模块化，管理和修改代码很方便，同时程序的可读性也会大大提高。C 语言头文件中也包含大量的库函数，可供用户直接使用来完成特定的功能。函数是 C 语言的重点，读者应认真掌握函数的概念及其应用。



9.7 习题

一、选择题

1. 下列程序的运行结果为 ()。

```
f()
{
    int x=1;
    x=x+2;
    printf("%d",x);
}
void main()
{
    f();
    f();
}
```

A. 22

B. 33

C. 11

D. 35

【提示】上述程序定义一个函数 `f()`，其中 `x` 为局部变量。`x` 的作用域只在函数体内，其生存期为函数调用时至函数调用结束。

2. 以下程序运行结果为 ()。

```
void fun(int m,int n)
{
    m=n;
    n=m;
}
void main()
{
    int x=5,y=7;
    fun(x,y);
    printf("%d,%d",x,y);
}
```




A. 7, 5

B. 5, 7

C. 5, 5

D. 7, 7

【提示】fun()函数是传值调用，因此不会影响主函数中 x 和 y 的值。

3. 以下程序运行结果为 ()。

```
void fun(int *m,int *n)
{
    *m=*n;
    *n=*m;
}
void main()
{
    int x=5,y=7;
    fun(&x,&y);
    printf("%d,%d",x,y);
}
```

A. 7, 5

B. 5, 7

C. 5, 5

D. 7, 7

【提示】上述程序 fun()是传址调用，因此形参值的改变会影响实参的值。

4. 以下程序的运行结果为 ()。

```
void f(int *s)
{
    s="Hello World! ";
}
void main()
{
    char s[]="Hello C! ";
    f(s);
    printf("%s",s);
}
```

A. Hello World!

B. Hello C!

C. Hello World

D. Hello C

【提示】本题 f()函数为传址调用，因此字符数组 s 中的内容会被改变为“Hello World!”。

5. 以下程序的运行结果为 ()。

```
void f(int a[])
{
    int i,s=0;
    for(i=0;i<5;i++)
        s=s+a[i];
    printf("%d",s);
}
void main()
```



```
{  
    int a[5]={1,3,5,7,9};  
    f(a);  
}
```

A. 20

B. 21

C. 22

D. 25

【提示】本题将数组 a 作为实参传递给 f() 函数，计算其累加和后输出。

6. 以下程序的运行结果为 ()。

```
void f(int *a,int *b)  
{  
    int t;  
    t=*a;  
    *a=*b;  
    *b=t;  
}  
void main()  
{  
    int x=3,y=4;  
    f(&x,&y);  
    printf("%d,%d",x,y);  
}
```

A. 3, 4

B. 4, 3

C. 3, 3

D. 4, 4

【提示】本题将 x 和 y 的地址作为实参传递给函数 f(), 在 f() 函数中交换了这两个数的值。

7. 以下程序的运行结果为 ()。

```
void f(int a[])  
{  
    int s=1,i;  
    for(i=0;i<5;i++)  
        s=s*a[i];  
    printf("%d",s);  
}  
void main()  
{  
    int x[5]={1,2,3,4,5};  
    f(x);  
}
```

A. 120

B. 100

C. 110

D. 140

【提示】本题将数组 x 作为参数传递给函数 f(), 然后在函数中计算每个元素的乘积后输出。



二、填空题

1. 现有以下函数，则其运行结果为_____。

```
void f()
{
    int i,s=0;
    for(i=0;i<5;i++)
        s=s+i;
    printf("%d",s);
}
```

【提示】上述程序通过 for 循环计算 0~4 的累加和保存至变量 s 中，然后输出变量 s 的值。

2. 以下程序的运行结果为_____。

```
int s(int x)
{
    x=x*2;
    return(x);
}
void main()
{
    int a=10,b;
    b=s(a);
    printf("%d,%d",a,b);
}
```

【提示】上述程序 s() 函数将 2x 赋值给 x 并返回 x 值，再在主函数 main() 中输出 a 和 b 的值。

3. 下列程序的运行结果为_____。

```
int f(int x[])
{
    int i;
    for(i=0;i<4;i++)
    {
        x[i]=x[i]+5;
        printf("%d, "x[i]);
    }
}
void main()
{
    int x[4]={1,2,3,4};
    int i;
    for(i=0;i<4;i++)
        printf("%d, "x[i]);
    printf("\n");
    f(x);
    for(i=0;i<4;i++)
```



```
printf("%d", x[i]);  
}
```

【提示】上述程序函数 f() 的功能为将数组 x 中的每一个元素值加上 5 后再输出。

4. 下列程序的运行结果为_____。

```
void f()  
{  
    static int a=0;  
    a=a+2;  
    printf("%d",a);  
}  
void main()  
{  
    f();  
    f();  
    f();  
}
```

【提示】变量 a 的类型为静态类型，因此变量 a 会始终存在，直到程序结束为止。

5. 下列程序的执行结果为_____。

```
int x=1;  
void f()  
{  
    int y;  
    y=x++;  
    printf("%d",y);  
}  
void main()  
{  
    f();  
    printf("%d",x);  
}
```

【提示】本题中 x 为全局变量，函数 f() 中 x 的值加了 1，因此 x 的值变化为 2。

6. 以下程序运行结果为_____。

```
void sort(int s[],int n)  
{  
    int i,j,t;  
    for(i=0;i<n-1;i++)  
        for(j=i+1;j<n;j++)  
            if(s[i]>s[j])  
            {  
                t=s[i];  
                s[i]=s[j];  
                s[j]=t;  
            }  
}
```



```
    for(i=0;i<9;i++)
    }
}
void main()
{
    int a[]={1,4,3,6,8,6,9,10,5};
    int n=5,i;
    sort(a,n);
    printf("%d",a[i]);
}
```

【提示】本题调用 `sort()` 函数对数组 `a` 进行排序，再输出数组中的每个元素。

7. 下列程序的执行结果为_____。

```
int a=10,b=20;
void change(int x,int y)
{
    int t;
    t=x;
    x=y;
    y=t;
}
void main()
{
    int c=30,d=40;
    change(a,b);
    change(c,d);
    printf("%d,%d,%d,%d",a,b,c,d);
}
```

【提示】变量 `a` 和 `b` 为全局变量，变量 `c` 和 `d` 为局部变量，因此调用 `change()` 函数之后，`a` 和 `b` 的值会改变，而 `c` 和 `d` 的值不会改变。

8. 下列程序的运行结果为_____。

```
int s=5;
void fun(int x)
{
    s=s-x;
}
void main()
{
    int i=6;
    fun(i);
    printf("%d,%d",i,s);
}
```



【提示】本题中 fun()函数中将 s-x 的值赋值给变量 s，然后输出变量 i 和 s 的值。

9. 以下程序运行结果为_____。

```
#include <math.h>
void main()
{
    int x=-5;
    int y=fabs(x);
    printf("%d,%d",x,y);
}
```

【提示】上述程序 x 的值为-5，然后调用 fabs()函数将 x 的绝对值赋值给变量 y，输出变量 x 和 y 的值。

10. 以下程序运行结果为_____。

```
void main()
{
    int t,a=1,b=4;
    a=2*b-1;
    t=a+b--;
    printf("%d,%d,%d",a,b,t);
}
```

【提示】上述程序将 2b-1 的值赋给变量 a，将 a+b--的值赋给变量 t，最后输出变量 a、b 和 t 的值。

11. 以下程序运行结果为_____。

```
int f(int a,int b)
{
    int c=5;
    return(a+b-5);
}
void main()
{
    int x=7,y=6;
    int z=f(x,y);
    printf("%d",z);
}
```

【提示】函数 f()的功能为返回 a+b-5 的值，在主函数中进行输出。

12. 以下程序运行结果为_____。

```
void f(int x,int *y)
{
    x=5;
    *y=*y+x;
}
void main()
```



```
{
    int x=3,y=4;
    f(x,&y);
    printf("%d,%d",x,y);
}
```

【提示】本题将变量 x 及变量 y 的地址传递给函数 f(), 执行 f() 函数再输出变量 x 和 y 的值。

13. 以下程序运行结果为_____。

```
int f(int x)
{
    return x++;
}
void main()
{
    int a=5,s=0;
    s=s+f(a);
    printf("%d,%d",a,s);
}
```

【提示】本题将变量 a 传递给函数 f(), 返回 x++ 的值, 输出变量 a 和 s 的值。

14. 以下程序运行结果为_____。

```
void fun(int a[],int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%d",++a[i]);
}
void main()
{
    int x[]={1,2,3,4,5,6,7};
    fun(x,5);
}
```

【提示】上述程序中 f() 函数将数组 x 中的每个元素值都加 1, 然后进行输出。

15. 以下程序运行结果为_____。

```
void fun(char s[],int n)
{
    int i;
    for(i=n-1;i>=0;i--)
        s[i]=s[i+1];
    printf("%s",s);
}
void main()
{
    char s[50]= "abcdefgh";
}
```



```
int n=6;
fun(s,n);
}
```

【提示】上述程序中 n 值为 6，将字符数组 s 和 n 传递给函数，在函数 $\text{fun}()$ 中进行相应的操作再输出。

三、编程题

1. 编写一个函数，从键盘输入一个字符串，将其反序输出。

【提示】利用字符串数组保存从键盘输入的字符串，要将字符串反序输出只要把字符串中的字符倒着输出即可。

【核心代码】

```
void main()
{
    gets(s);
    f(s);
}
void f(char *s)
{
    i=strlen(s);
    for(j=i;j>=0;j--)
        printf("%c",s[j]);
}
```

2. 从键盘输入一个数 n ，求 $1+2+3+\cdots+n$ 的值，要求用递归的方式求解。

【提示】本题要求采用递归的方式，可以用递归函数来求解。设置一个累加器 s 和变量 n ， s 初始值为 0，再利用递归将 n 至 1 的值累加和赋给变量 s ，最后输出变量 s 即为计算得出的结果。

【核心代码】

```
int s=0;
void fun(int n)
{
    s=s+n;
    n--;
    if(n==0)
        break;
    else
        f();
}
void main()
{
    scanf("%d",&n);
    fun(n);
}
```




3. 从键盘输入一个字符串, 要求从该字符串的第 n 个位置开始至末尾组成一个新的字符串并将结果输出至屏幕。

【提示】定义一个字符指针 p , 将数字 n 作为实参传递给函数。函数通过 n 移动指针的位置, 使指针指向字符串的第 n 个位置, 然后将字符串赋值给另一个字符数组直至字符串的末尾为止。

【核心代码】

```
void f(char *p)
{
    for(i=0;i<n-1;i++)
        p++;
    while(*p!= '\0')
        q++=p++;
    printf("%s",q);
}
```

4. 从键盘输入一个字符串, 统计其中字母的个数。

【提示】利用字符数组保存输入的字符串, 同时设置一个计数器, 其初始值为 0。对字符数组中的字符一一进行比较, 若为字母则计算器加 1, 否则不变。最后输出计数器的值即为字母的个数。

【核心代码】

```
void f(char *s)
{
    while(*s!= '\0')
    {
        if((*s>='a'&&*s<='z') || (*s>='A'&&*s<='Z'))
            count++;
        s++;
    }
}
```

5. 编写一个函数, 判断一个数是否为素数。

【提示】素数是指除了 1 和本身之外, 不能被其他数整除的数。因此可以利用 for 循环穷举进行判断, 若一个数 n 不能被 $2 \sim n-1$ 中的数整除, 则该数为素数。

【核心代码】

```
void fun(int n)
{
    flag=1;
    for(i=2;i<=n/2;i++)
        if(n%i==0)
            flag=0;
    if(flag==1)
        printf("该数为素数");
    else
        printf("该数不是素数");
}
```



6. 从键盘输入多个字符串, 编写一个函数对这些字符串从大到小排序, 并输出排序结果。

【提示】从键盘输入 5 个字符串可以利用 `gets()` 函数或 `scanf()` 函数来实现。对 5 个字符串排序则可以用冒泡排序或选择排序法, 最后利用 `printf()` 或 `puts()` 函数输出排好序的结果。

【核心代码】

```
void f(char **s)
{
    for(i=0;i<4;i++)
        for(j=i+1;j<5;j++)
            if(strcmp(s[i],s[j])<0)
            {
                t=s[i];
                s[i]=s[j];
                s[j]=t;
            }
    for(i=0;i<5;i++)
        printf("%s\n",s[i]);
}
```

7. 编写一个程序, 求 $1!+2!+3!+4!$ 的值。

【提示】求 $1!+2!+3!+4!$ 的值, 可以将其单独写成一个函数, 然后在主函数中调用该函数即可。

【核心代码】

```
void fun()
{
    s=0;
    for(i=1;i<=4;i++)
    {
        x=x*i;
        s=s+x;
    }
}
```

8. 编写一个函数, 判断一个字母是否为小写字母, 若为小写字母则转化成大写字母后输出。

【提示】判断一个字母是否为小写字母, 可以用该字符是否大于或等于字符 `a` 并且小于或等于字符 `z` 进行判断。若在该范围内则为小写字母, 否则不是。

【核心代码】

```
void f(char c)
{
    if(c>='a'&&c<='z')
        printf("该字母为小写字母\n");
    else
        printf("该字母为大写字母\n");
}
```



9. 编写一个函数，将字符串逆序输出至屏幕。

【提示】将字符串逆序输出，可以使指针先指向字符串的末尾，然后向前移动指针并输出直到字符串的起始位置为止。

【核心代码】

```
void nixu(char *p)
{
    p1=p;
    while(*p1!= '\0')
        p1++;
    p1--;
    while(1)
    {
        putchar(p1);
        p1--;
        if(p1==p)
            break;
    }
}
```

10. 编写一个函数，实现字符串的连接功能。

【提示】将一个字符串连接到另一个字符串后，可以通过指针进行操作。指针 p1 指向一个字符串的末尾，指针 p2 指向另一个字符串开头，然后将字符串连接到另一个字符串上即可。

【核心代码】

```
void s(char *p1,char *p2)
{
    while(*p1!= '\0')
        p1++;
    while(*p2!= '\0')
        *p1++=*p2++;
    *p1='\0';
}

void main()
{
    s(s1,s2);
}
```

第 10 章 结构型、共用型、枚举型 及用户自定义型数据

通过前面的学习可以了解，变量、数组一旦定义后只能存储定义的类型数据，例如一个整型数组，数组中的元素都为整型。但在实际生活中，每一组数据中的元素可能不是同一类型的数据，例如要记录一个人的信息，包含姓名、性别、年龄、职业等方面的信息，这就不能用数组来存储了。

C 语言提供了另外一种类型结构体来解决这类问题。它把一个人的信息包含成一个整体，即一个结构体变量可以包含不同数据类型的数据。结构体类型是 C 语言中重要的构造数据类型，可以很方便地对不同数据类型的数据进行操作。

本章主要涉及的知识点有：

- 结构体类型概念及其应用；
- 共用体类型；
- 枚举类型；
- 结构体数组；
- 结构体指针；
- typedef 定义。



10.1 结构体类型

结构体通过将不同的数据类型封装成一个整体结构，再对其中的数据进行引用。人们可以根据实际情况来构造不同的结构类型。在本节将讲解结构体类型的定义及其应用。

10.1.1 结构体类型简介

结构体类型是不同类型数据的集合，它是用户自己定义的数据类型。当要记录一个学生的信息时，包含学号、性别、名字等数据，单独定义数组记录则显得很麻烦。通过结构体类型可以把这些信息整合成一个整体，处理起来也很方便，程序的可读性也大大增强。

结构体类型与数组主要包括以下两个方面的区别：

(1) 结构体中可以包含不同类型的数据，数组只能包含同一类型的数据。(2) 结构体可以相互赋值，而数组不能直接相互赋值。因为结构体类型本身是数据类型，而数组是同一类型数据的集合。



10.1.2 结构体类型定义

在 C 语言中，一个学生的信息可以用如下结构体保存：

```
struct s
{
    char name[10];
    int age;
    char sex;
};
```

上述语言定义了一个结构体类型 `s`，其中 `struct` 为 C 语言中结构体类型的关键字，`s` 为结构体名，大括号中为一个结构体包含的成员，其中包括字符型数组 `name`，整型变量 `age` 和字符型变量 `sex`。根据实际情况的不同，结构体类型中的成员可以改变。

结构体定义形式如下：

```
struct 结构体名
{
    结构体成员
};
```

结构体成员可以是变量、数组、指针等数据类型。定义了结构体之后，就可以用它来声明结构体变量。结构体变量声明后，即可引用结构体中的成员。

结构体变量的定义方式有以下三种：

(1) 先定义结构体，后定义结构体变量。例如：

```
struct s
{
    char name[10];
    int age;
    char sex;
};
struct s s1,s2;
```

通过结构体 `s` 定义了两个结构体变量 `s1` 和 `s2`，程序可以直接引用两个结构体变量中的成员。

(2) 定义结构体同时定义结构体变量。例如：

```
struct s
{
    char name[10];
    int age;
    char sex;
}s1,s2;
```

在定义结构体 `s` 的同时定义了两个结构体变量 `s1` 和 `s2`。



(3) 只定义结构体变量。例如：

```
struct
{
    char name[10];
    int age;
    char sex;
}s1,s2;
```

上述程序中没有定义结构体类型名，直接定义了两个结构体变量 s1 和 s2。因为没有定义结构体类型名，因此除了直接定义，不能使用其他方法定义结构体变量。



注意：

(1) 结构体是用户自定义类型，结构体名可用来定义结构体变量，但不能用来传递信息。

(2) 结构体中可以嵌套另外一个结构体，即一个结构体中可以包含多个结构体。

例如以下程序：

```
struct time
{
    int hour;
    int minute;
    int second;
};
struct date
{
    int day;
    int month;
    int year;
    struct time t;
}d;
```

在 date 结构体中嵌套了 time 结构体，通过 date 结构体可以调用 time 结构体中的成员。

10.1.3 结构体类型引用

定义结构体变量后，可以通过操作符“.”来引用结构体中的成员。其引用方式如下：

结构体变量名.成员名；

例如：

```
struct s
{
    char name[10];
    int age;
    char sex;
```



```
}s1;
```

若有以上结构体，要将 s1 中 age 赋值为 20，则可用如下语句实现：

```
s1.age=20;
```

【范例 10.1】利用结构体，从键盘输入一个学生的信息，将这些信息存储至结构体中并输出。

分析：定义一个结构体，包含学生的一些基本信息。然后对结构体进行赋值，最后输出结构体中的成员即可。

范例 10.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      struct s                                        /*定义结构体类型变量 s1*/
05      {
06          char name[10];
07          int age;
08          char sex;
09          float score;
10      }s1;
11      printf("输入学生姓名: ");
12      gets(s1.name);                                  /*从键盘获取字符串保存至结构体相应变量中*/
13      printf("输入学生性别: ");
14      s1.sex=getchar();
15      printf("输入学生岁数: ");
16      scanf("%d",&s1.age);
17      printf("输入学生分数: ");
18      scanf("%f",&s1.score);
19      printf("名字:%s\n",s1.name);                    /*输出该名学生的信息至屏幕*/
20      printf("年龄:%d\n",s1.age);
21      printf("性别:%c\n",s1.sex);
22      printf("分数:%f\n",s1.score);
23  }
```

【代码分析】本例为结构体的简单范例，详细代码分析如下：

- 第 4~10 行，定义了一个结构体变量 s1，其中包括姓名、年龄、性别、分数等信息。
- 第 11~18 行，通过 scanf() 函数实现输入学生的姓名、性别、年龄、分数等信息。

【运行结果】该程序的执行结果如图 10-1 所示。



注意：第 10 行后面的分号一定不能省去，结构体大括号后的分号为结构体定义基本形式，若没加上则会出错。

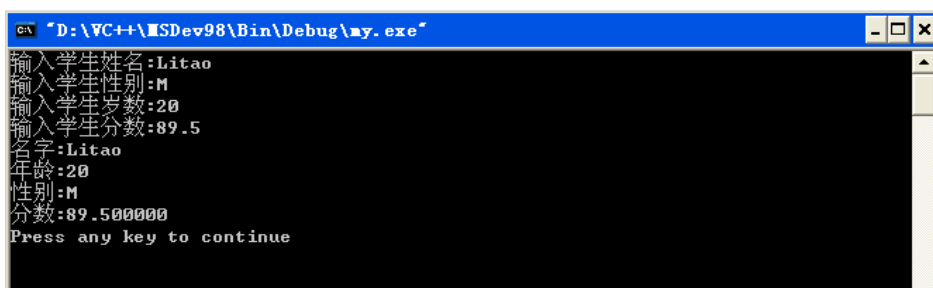


图 10-1 范例 10.1 结果图

对于嵌套结构体，例如下面的结构体：

```
struct time
{
    int hour;
    int minute;
    int second;
};
struct date
{
    int day;
    int month;
    int year;
    struct time t;
}d;
```

要引用 time 结构体成员，可以用多级操作符“.”来引用，如 d.t.hour、d.t.minute 和 d.t.second。

10.1.4 结构体变量初始化

结构体变量与变量相似，在定义结构体变量的同时便可对其各个成员赋初始值，即结构体变量的初始化。在初始化时，用户所赋的值应与结构体中的成员类型一一对应，不能颠倒次序，否则初始化会出错。

结构体变量初始化与定义类似，也有三种方式。

(1) 第一种方式：

```
struct s
{
    char name[10];
    int age;
    char sex;
    float score;
};
struct s s1={"LiHai",20, 'M',90.5};
```

(2) 第二种方式：



```
struct s
{
    char name[10];
    int age;
    char sex;
    float score;
}s1={"LiHai",20, 'M',90.5};
```

(3) 第三种方式:

```
struct
{
    char name[10];
    int age;
    char sex;
    float score;
}s1={"LiHai",20, 'M',90.5};
```

上述三种方式都可以实现结构体变量的初始化，其效果都是一样的。



注意：不能在结构体内部进行初始化。例如：

```
struct
{
    char name[10]= "LiHai";
    int age=20;
    char sex='M';
    float score=90.5;
}s1;
```



10.2 结构体数组

当要处理多个人的信息时，就需要采用数组。而每个人信息的记录要采用结构体类型，因此 C 语言提供了结构体数组来解决此类问题。在本节中将讲解结构体数组的定义、初始化及其应用。

10.2.1 结构体数组定义

结构体定义形式与数组定义形式类似，它包含以下两种定义方式：

(1) 先定义结构体，后定义结构体数组。

```
struct s
{
    char name[10];
    int age;
```



```
char sex;
float score;
};
struct s a[10];
```

(2) 定义结构体的同时定义结构体数组。

```
struct s
{
    char name[10];
    int age;
    char sex;
    float score;
} a[10];
```

上述程序中定义了包含 10 个元素的结构体数组，每一个结构体元素占据 17 个字节，其中字符型数组占 10 个字节，整型变量 `age` 占 2 个字节，字符型变量 `sex` 占 1 个字节，浮点型变量 `score` 占 4 个字节。

10.2.2 结构体数组引用

结构体数组定义以后，通过下标即可引用相应的结构体元素。接下来通过下面的范例来了解结构体数组引用方式。

【范例 10.2】 现有 3 个候选人，10 个人进行投票，统计每一个候选人的票数。

分析：可以将这 3 个候选人定义为结构体类型，10 个人进行投票，将每个人的投票与结构体中的名字进行比较，若相等则其票数加 1，最后输出 3 个候选人的票数。

范例 10.2 代码实现

```
01  #include <stdio.h>
02  #include <string.h>           /*包含 string.h 头文件*/
03  struct                        /*定义结构体变量数组 p 并进行初始化*/
04  {
05      char name[10];
06      int c;
07  }p[3]={ "zhang",0, "li",0, "hu",0};
08  void main()
09  {
10      int i,j;
11      char n[10];
12      for(i=0;i<10;i++)
13      {
14          scanf("%s",n);        /*从键盘输入候选人名字*/
15          for(j=0;j<3;j++)
16              if(strcmp(n,p[j].name)==0) /*与候选人名单进行比较，若相等则其票数加 1*/
17              p[j].c++;
```



```
18     }
19     printf("最终结果:\n");
20     for(j=0;j<3;j++)
21         printf("%s:%d 票\n",p[j].name,p[j].c);
22 }
```

【代码分析】本例为结构体应用范例，详细代码分析如下：

- 第 4~7 行，定义了一个结构体数组 `p`，并对其赋值进行了初始化。
- 第 12~17 行，实现投票功能。外层循环执行 10 次，表示共有 10 个人进行投票，将每个人的投票与候选人进行比较，若相等则其票数加 1。

【运行结果】该程序的执行结果如图 10-2 所示。

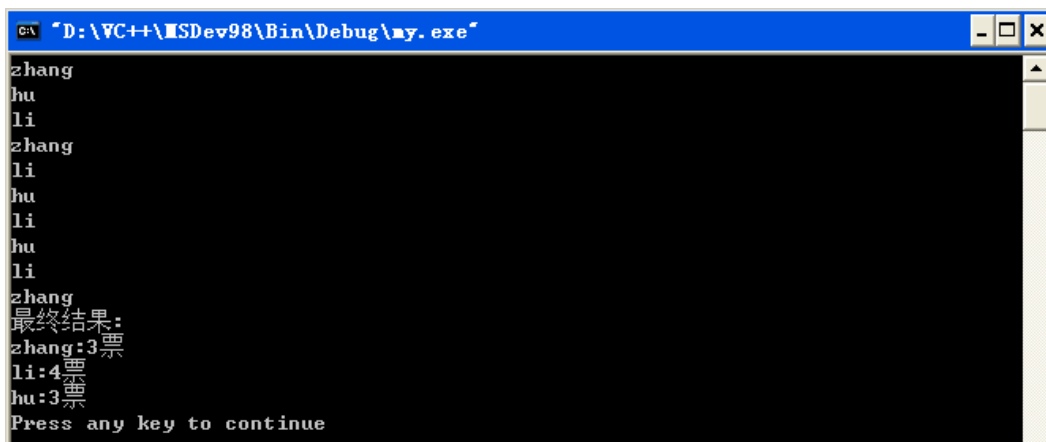


图 10-2 范例 10.2 结果图

10.2.3 结构体数组初始化

结构体数组可以在定义时就赋予初始值，即进行初始化。结构体数组初始化方式有两种，如下所示。

(1) 初始化数组所有元素。例如：

```
struct
{
    char name[10];
    int age;
    float socre;
}s[3]={ { "Zhang Lin",20,80.5},{ "Li Tao",19,89.0},{ "Xiu Yan",21,90.0}};
```

定义了一个包含三个元素的结构体数组，并对其中所有元素进行了初始化。

(2) 初始化数组部分元素，且结构体中嵌套另一个结构体。例如：

```
struct Time
{
    int hour;
    int minute;
```



```
int second;
struct Date
{
    int year;
    int month;
    int day;
}
}t[10]={
{6,56,34,{2009,5,24}},          /*初始化第一个元素*/
{7,35,56,{2010,7,2}};
};
```



10.3 结构指针

在前面的章节中讲解过指针的概念，指针是用来指向变量地址的。结构指针指向已定义的结构体变量在内存中的首地址，从而对地址中的内容进行操作。在本节中将讲解结构指针的概念及其应用。

10.3.1 结构体指针概念及其定义

结构体指针可以指向结构体变量的首地址，其定义形式如下：

```
struct 结构体类型 *指针名;
```

例如以下程序：

```
struct st
{
    char name[10];
    int age;
    char sex;
    float score;
}s;
struct st *p;
```

定义了一个结构体 st 的指针 p，但其值是不确定的。通过以下语句可将结构体变量 s 的地址赋给 p，使 p 指向结构体变量 s。

```
p=&s;
```

其中“&”为取地址符，&s 表示结构体变量 s 的地址。指针赋值后，通过指针即可访问结构体中的成员，如：

```
(*p).name    (*p).sex
```

其中括号不能省去，因为“.”的优先级比“*”高。在 C 语言中通过指针还可以用另外一



种方式来访问结构体成员，即使用运算符“->”，如下所示。

```
p->name    p->age    p->sex
```

【范例 10.3】通过下面的例子，初步了解结构体指针。

分析：要通过结构体指针对结构体进行操作，首先应将结构体变量的地址赋给结构指针，使指针指向相应的结构体。

范例 10.3 代码实现

```
01  #include <stdio.h>
02  struct st                                /* 自定义结构体类型 */
03  {
04      char name[10];
05      int age;
06      float score;
07  };
08  void main()
09  {
10      struct st s1={"Zhang Tao",21,89.5},s2={"Li Tu",22,87.0};
11      struct st *p;
12      p=&s1;                                /* 使指针 p 指向结构体变量 s1 */
13      printf("%s,%d,%.1f\n",p->name,p->age,p->score);    /* 输出该结构体信息 */
14      p=&s2;
15      printf("%s,%d,%.1f\n",(*p).name,(*p).age,(*p).score);
16      p->age=24;                             /* 修改结构体信息 */
17      printf("%s,%d,%.1f\n",(*p).name,(*p).age,(*p).score);
18  }
```

【代码分析】本例为结构体指针简单范例，详细代码分析如下：

- 第 11 行，定义了一个结构体指针 p。
- 第 12 行，将 s1 的首地址赋给指针 p，使指针 p 指向结构体 s1。
- 第 16 行，利用指针改变 s2 结构体中的成员值。

【运行结果】该程序的执行结果如图 10-3 所示。

```
C:\ "D:\VC++\MSDev98\Bin\Debug\my.exe"
Zhang Tao,21,89.5
Li Tu,22,87.0
Li Tu,24,87.0
Press any key to continue_
```

图 10-3 范例 10.3 结果图

10.3.2 结构体数组指针

结构体数组也可以定义指针，通过指针对其进行操作，例如下面的例子。



【范例 10.4】编写一个程序，通过结构体数组指针对结构体数组进行操作。

分析：结构体数组指针指向结构体数组，通过改变指针的值访问结构体数组中的元素。

范例 10.4 代码实现

```
01  #include <stdio.h>
02  struct st                               /*定义结构体数组 s*/
03  {
04      char name[10];
05      int age;
06      char sex;
07      float score;
08  }s[3];
09  void main()
10  {
11      int i;
12      struct st s[3]={                     /*对结构体数组 s 进行初始化*/
13          {"Zhang Tao",20, 'M',90.0},
14          {"Li Tu",19, 'W',89.5},
15          {"Wang Yan",20, 'W',98.5},
16      };
17      struct st *p;
18      p=s;
19      for(i=0;i<3;i++)                     /*通过指针 p 输出数组 s 中的每个元素信息*/
20      {
21          printf("%s,%d,%c,%.1f\n",p->name,p->age,p->sex,p->score);
22          p++;
23      }
24  }
```

【代码分析】本例为结构体数组指针应用范例，详细代码分析如下：

- 第 2~8 行，定义了一个结构体数组 s，其中包括 4 个结构体成员。
- 第 19~23 行，通过结构体数组指针和 for 循环输出数组中每一个结构体成员的值。

【运行结果】该程序的执行结果如图 10-4 所示。

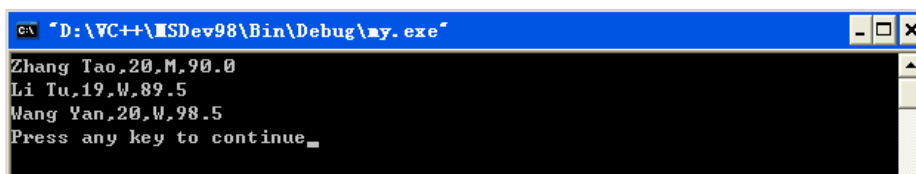


图 10-4 范例 10.4 结果图

在使用结构体数组指针时，应注意以下两个方面。

(1) (++p) ->num 和 (p++) ->num 的区别：(++p) ->num：先使 p 的值加 1，使 p 指向数组的下一个元素，再访问该元素的结构体成员 num。(p++) ->num：先访问元素的结构体成员 num，再使 p 的值加 1，指向下一个元素。



(2) 结构体数组指针 `p` 指向数组有以下三种方式：

```
p=s; p=&s; p=&s[0];
```

上述三种方式中，一般使用第一种，因其简洁方便，后两种使用较少。

10.3.3 结构体指针应用

结构体指针在 C 语言中应用很广泛，可以简单快捷地对结构体进行操作。下面通过几个范例来看一下结构体指针的应用。

【范例 10.5】 通过结构体指针，实现结构体的输入、输出。

分析：定义一个结构指针，指向相应的结构体变量，通过 C 语言中的“.”或“->”运算符即可操作结构体变量。

范例 10.5 代码实现

```
01  #include <stdio.h>
02  #include <stdlib.h>           /*包含 stdlib.h 头文件*/
03  struct st
04  {
05      char name[10];
06      char age[5];
07      char sex[6];
08      float score;
09  }s;
10  void main()
11  {
12      struct st *p=&s;
13      char a[20];
14      printf("输入名字: ");
15      gets(p->name);             /*从键盘获取 p->name 的值*/
16      printf("输入年龄: ");
17      gets(p->age);
18      printf("输入性别: ");
19      gets(p->sex);
20      printf("输入分数: ");
21      gets(s);
22      p->score=atof(s);           /*将分数转化为浮点型赋值给 p->score*/
23      printf("名字:%s\n",p->name);
24      printf("性别:%s\n",p->sex);
25      printf("年龄:%s\n",p->age);
26      printf("分数:%.1f\n",p->score);
27  }
```

【代码分析】 本例为结构指针应用范例，详细代码分析如下：

- 第 12 行，定义了一个结构指针 `p`，指向结构体变量 `s`。



- 第 14~21 行, 实现结构体成员数据的输入。
- 第 22 行, 利用了 `stdlib.h` 头文件中的 `atof()` 函数对数据进行转化。该函数的功能是将字符型数据转化为浮点型数据。

【运行结果】该程序的执行结果如图 10-5 所示。

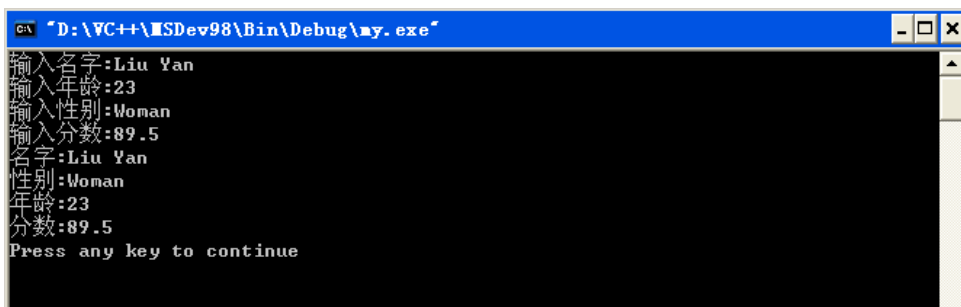


图 10-5 范例 10.5 结果图

【范例 10.6】通过结构体指针, 对文件进行读和写入操作。

分析: 对文件进行读写可以利用 `fread()` 和 `fwrite()` 函数来实现, 其中 `fread()` 函数用来读取文件中的数据, `fwrite()` 函数用来将数据写至相应的文件中。

先建立一个名为 `input.txt` 的文本文件, 存放 5 个人的信息, 如下所示。

```
Lihua 20 man 86.0
Zhangming 23 man 89.0
litao 19 man 87.5
liqiu 20 woman 90.5
wangyan 19 woman 92.0
```

然后编写程序读取 `input.txt` 中的内容, 写至 `output.txt` 文件中。

范例 10.6 代码实现

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  struct st
04  {
05      char name[10];
06      char age[5];
07      char sex[6];
08      float score;
09  };
10  void main()
11  {
12      struct st s[5], *p;
13      FILE *fp1, *fp2;          /*定义文件指针 fp1 和 fp2*/
14      int i;
15      p=s;
16      fp1=fopen("input.txt", "r"); /*以读的方式打开 input.txt 文件*/
```




```
17  fp2=fopen("ouput.txt","w");      /*以写的方式打开 ouput.txt 文件*/
18  for(i=0;i<5;i++)                  /*将 input.txt 中的内容写至结构体数组 s 中*/
19      fread(p++,sizeof(struct st),1,fp1);
20  p=p-5;
21  for(i=0;i<5;i++)                  /*将结构体数组 s 内容写至 ouput.txt 文件中*/
22      fwrite(p++,sizeof(struct st),1,fp2);
23  fclose(fp1);                      /*关闭文件指针 fp1*/
24  fclose(fp2);
25  }
```

【代码分析】本题通过结构指针对文件进行操作，详细代码分析如下：

- 第 13 行，定义了两个文件指针 `fp1` 和 `fp2`，这些有关文件的操作将会在后面的章节中讲到。
- 第 16、17 行，利用 `fopen()` 函数打开相应的文件。
- 第 19 行，将 `fp1` 指针指向的文件即 `input.txt` 中的数据读出，保存在结构体数组 `s` 中。
- 第 22 行，用 `fwrite()` 函数将结构体数组 `s` 中的内容写至另一个文件中。
- 第 23、24 行，调用 `fclose()` 函数，关闭文件的读和写操作。

【运行结果】程序运行结果如下所示。

```
Lihua 20 man 86.0
Zhangming 23 man 89.0
litao 19 man 87.5
liqiu 20 woman 90.5
wangyan 19 woman 92.0
```



10.4 结构与函数参数

结构体可以用来作为函数的参数，其中包括 4 个方面：结构体变量作为函数参数，结构体地址作为函数参数，结构体数组作为函数参数，结构体指针作为函数参数。在本节中将会重点讲解这些情况。

10.4.1 结构变量作为函数参数

将结构体变量作为实参传递给一个函数，则形参应与其具有相同的结构类型。这种方式实际上是将结构体变量的值传递给函数，即传值调用。

【范例 10.7】编写一个程序，将结构体变量作为实参传递给其他函数。

分析：结构体变量直接传递给函数，是传值调用，因此函数内形参值的变化不会影响实参的值。

范例 10.7 代码实现

```
01  #include <stdio.h>
```



```
02  #include <stdlib.h>
03  struct st
04  {
05      char name[10];
06      char age[5];
07      char sex[6];
08      float score;
09  };
10  void out(st stu)                /*自定义函数 out()*/
11  {
12      printf("名字:%s\n",stu.name);    /*输出结构体成员*/
13      printf("性别:%s\n",stu.sex);
14      printf("年龄:%s\n",stu.age);
15      printf("分数:%.1f\n",stu.score);
16  }
17  void in(st stu)
18  {
19      char score[10];
20      printf("输入名字: ");
21      gets(stu.name);                /*从键盘获取字符串至 stu.name 中*/
22      printf("输入年龄: ");
23      gets(stu.age);
24      printf("输入性别: ");
25      gets(stu.sex);
26      printf("输入分数: ");
27      gets(score);
28      stu.score=atof(score);          /*将 score 转化为浮点型赋值给 stu.score*/
29      out(stu);                      /*调用 out()函数输出学生信息*/
30  }
31  void main()
32  {
33      struct st s={"Li Ming","20","man",87.5};
34      in(s);                        /*调用 in()函数*/
35      out(s);
36  }
```

【代码分析】本例将结构体变量作为实参，详细代码分析如下：

- 第 10~16 行为自定义函数 out()。该函数的功能为输出结构体各个成员的值，其中 st 作为函数的形参用来接收实参的值。
- 第 17~30 行，用户自定义函数 in()，用来实现数据从键盘的输入。其中第 29 行函数内部嵌套了 out()函数，用来输出结构体变量的值。
- 第 35 行，在主函数中调用 out()函数，输出结构体变量 s 中各个成员的值。

【运行结果】该程序的执行结果如图 10-6 所示。



注意：虽然 in() 函数中对结构体变量重新进行了赋值，但由于传递方式为传值调用，因此不会影响主函数 main() 中结构体变量的值。

```

D:\VC++\MSDev98\Bin\Debug\z.exe
输入名字:Li lin
输入年龄:21
输入性别:woman
输入分数:90.0
名字:Li lin
性别:woman
年龄:21
分数:90.0
名字:Li Ming
性别:man
年龄:20
分数:87.5
Press any key to continue_

```

图 10-6 范例 10.7 结果图

10.4.2 结构体地址作为函数参数

将结构体变量地址作为参数传递给函数，则形参应为相同类型的指针。这种方式为传址方式，因此函数内结构体变量值的变化会影响实参。

【范例 10.8】编写一个程序，将结构体变量地址作为实参传递给其他函数。

分析：结构体变量地址传递给函数，是传址调用，因此形参值的变化会影响实参的值。

范例 10.8 代码实现

```

01  #include <stdio.h>
02  #include <stdlib.h>
03  struct st
04  {
05      char name[10];
06      char age[5];
07      char sex[6];
08      float score;
09  };
10  void out(st stu)
11  {
12      printf("名字:%s\n",stu.name);
13      printf("性别:%s\n",stu.sex);
14      printf("年龄:%s\n",stu.age);
15      printf("分数:%.1f\n",stu.score);
16  }
17  void in(st *stu)
18  {
19      char score[10];
20      printf("输入名字: ");

```



```

21     gets(stu->name);                      /*从键盘获取数据至 stu->name 中*/
22     printf("输入年龄: ");
23     gets(stu->age);
24     printf("输入性别: ");
25     gets(stu->sex);
26     printf("输入分数: ");
27     gets(score);
28     stu->score=atof(score);                /*将分数转化为浮点型保存至 stu->score 中*/
29 }
30 void main()
31 {
32     struct st s={"Li Ming","20","man",87.5};
33     out(s);                               /*调用函数 out()*/
34     in(&s);                               /*调用 in() 函数并将变量 s 地址传递给函数*/
35     out(s);
36 }

```

【代码分析】本例将结构体变量地址作为实参，详细代码分析如下：

- 第 33~35 行，调用 out()和 in()函数进行数据的输入、输出。其中 in()函数形参为指针类型，因此实参为结构体变量 s 的地址。

【运行结果】该程序的执行结果如图 10-7 所示。

图 10-7 范例 10.8 结果图

10.4.3 结构体数组作为函数参数

在前面的章节中讲解过，数组名可以代表数组的首地址。因此结构体数组可以作为参数传递给函数，即将结构体数组的首地址传递给函数。

【范例 10.9】编写一个程序，将结构体数组作为实参传递给其他函数。

分析：结构体数组传递给函数，即将结构体数组首地址传递给函数，然后便可对数组中的内容进行操作。

范例 10.9 代码实现

```
01 #include <stdio.h>
```



```
02  #include <stdlib.h>
03  struct st
04  {
05      char name[10];
06      char age[5];
07      char sex[6];
08      float score;
09  };
10  void prin(st s[])                /*自定义函数 prin()*/
11  {
12      int i;
13      for(i=0;i<3;i++)
14      {
15          printf("名字:%s\n",s[i].name);
16          printf("年龄:%s\n",s[i].age);
17          printf("性别:%s\n",s[i].sex);
18          printf("分数:%f\n",s[i].score);
19      }
20  }
21  void main()
22  {
23      struct st s[3]={             /*定义结构体数组 s 并进行初始化*/
24          {"Li Ming", "20", "man", 87.0},
25          {"Hu Su", "19", "woman", 89.0},
26          {"Jun Yi", "21", "man", 90.5}
27      };
28      prin(s);
29  }
```

【代码分析】本例将结构体数组作为实参，详细代码分析如下：

- 第 10~20 行，用户自定义函数 prin()，其中结构体数组 s 作为函数的形参。
- 第 28 行，调用 prin() 函数，结构数组 s 作为实参传递给函数。

【运行结果】该程序的执行结果如图 10-8 所示。

```
名字:Li Ming
年龄:20
性别:man
分数:87.000000
名字:Hu Su
年龄:19
性别:woman
分数:89.000000
名字:Jun Yi
年龄:21
性别:man
分数:90.500000
Press any key to continue_
```

图 10-8 范例 10.9 结果图



10.5 共用体

共用体与结构体类似，都是将不同的数据类型组合在一起，但是共用体与结构体所占的空间不同。在本节中将讲解共用体的概念及其应用。

10.5.1 共用体概念及其定义

共用体为用户自定义构造类型，但与结构体类型不同，结构体类型存储时采用覆盖方式，即后一个类型的数据会覆盖前一个类型数据。结构体变量所占空间为所有成员占据的空间之和，而共用体变量所占空间为成员中所占空间最大的成员字节数。

共用体定义形式如下：

```
union 共用体名
{
    成员表
}共用体变量名;
```

例如：

```
union day
{
    int x;
    float y;
    double z;
    char c;
}t;
```

定义了一个共用体类型和共用体变量 `t`，其中包括整型数据 `x`，单精度浮点型数 `y`，双精度浮点型 `z`，字符型 `c`，共用体变量 `t` 在内存占据 8 个字节。

与结构体类型类似，对共用体变量进行操作，可以通过“.”和“->”运算符来实现。若操作对象为共用体变量，则用“.”运算符，若操作对象为共用体类型指针，则使用“->”运算符。

共用体与结构体类似，但也有一些不同，主要包含以下三方面：

(1) 结构体成员都有自己的存储空间，结构体变量所占存储空间为各个成员所占存储空间之和。共用体成员共享一片存储空间，共用体变量所占存储空间为各个成员中占存储最大的成员的字节数。(2) 结构体变量可以同时对其所有成员赋值，而共用体类型不能同时对所有成员赋值，只能单独赋值。(3) 结构体变量可以初始化，而共用体在定义时不能赋值。

10.5.2 共用体变量应用

要引用共用体变量中的成员，可以通过共用体变量来引用，也可以通过共用体指针来引用。共用体具有以下特点：

(1) 共用体变量的值为最后一次赋的值。例如：

```
union day
```



```
{
    char c;
    int i;
    float j;
}t;
t.c='f';
t.i=6;
t.j=4.5;
```

最后共用体变量 `t` 的值为 4.5，前面两个值都被覆盖了。

(2) 共用体变量在定义时不能赋值，即初始化。例如：

```
union day
{
    char c;
    int i;
    float j;
}t={'z',4,9.0};
```

上述程序是错误的。

(3) 不能对共用体变量整体进行赋值。例如：

```
union day
{
    char c;
    int i;
    float j;
}t;
t='w';
t=98;
```

最后两行都是错误的。

(4) 与结构体类似，可以定义数组。例如：

```
union day
{
    char c;
    int i;
    float j;
}t[5];
```

【范例 10.10】 通过下面的例子，了解共用体的存储过程。

分析：共用体是用户自定义类型，是不同数据类型的集合。

范例 10.10 代码实现

```
01  #include <stdio.h>
02  union                                /*定义共用体变量 s*/
```



```
03  {
04    int x;
05    char y;
06    float z;
07  }s;
08  void main()
09  {
10    printf("sizeof(s.x)=%d\n",sizeof(s.x));          /*输出 s.x 所占存储空间*/
11    printf("sizeof(s.y)=%d\n",sizeof(s.y));
12    printf("sizeof(s.z)=%d\n",sizeof(s.z));
13    printf("sizeof(s)=%d\n",sizeof(s));
14  }
```

【代码分析】本例为共用体简单范例，详细代码分析如下：

- 第 2~7 行，定义了一个共用体变量 s，其中包括 x、y、z 三个成员。
- 第 10~13 行，利用 sizeof() 计算共用体变量及其成员所占存储空间，再利用 printf() 函数进行输出。

【运行结果】该程序的执行结果如图 10-9 所示。

图 10-9 范例 10.10 结果图

10.5.3 共同体与结构体的嵌套

在 C 语言中，结构体还可以与共用体嵌套使用，例如下面的例子。

【范例 10.11】编写一个程序，实现结构体与共用体的嵌套使用。

分析：结构体可以嵌套共用体，共用体中也可以嵌套结构体。当要引用其中的成员时，可以通过“.”或“->”运算符来实现。

范例 10.11 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04    struct std                                /*定义结构体变量 s*/
05    { union                                    /*内嵌共用体变量 a*/
06      {
07        int m;
08        int n;
09      }a;
```




```
10     int x;  
11     int y;  
12 }s;  
13     s.x=3;                                /*将 3 赋值给 s.x*/  
14     s.y=5;  
15     s.a.m=s.x*s.y;  
16     s.a.n=s.y-s.x;  
17     printf("结果:%d,%d\n",s.a.m,s.a.n);    /*输出 s.a.m 和 s.a.n 的值*/  
18 }
```

【代码分析】本例结构体中嵌套使用共用体，详细代码分析如下：

- 第 4~12 行，定义了一个结构体变量 s，其中嵌套了共用体变量 a。
- 第 15 行，将结构体成员 x 和 y 的乘积赋给共用体成员 m，其值为 15。
- 第 16 行，将结构体成员 y 与 x 的差值赋给共用体成员 n，值为 2。因为共用体是共用一块存储空间，因此会覆盖 m 的值，最终 m 和 n 的值都为 2。

【运行结果】该程序的执行结果如图 10-10 所示。

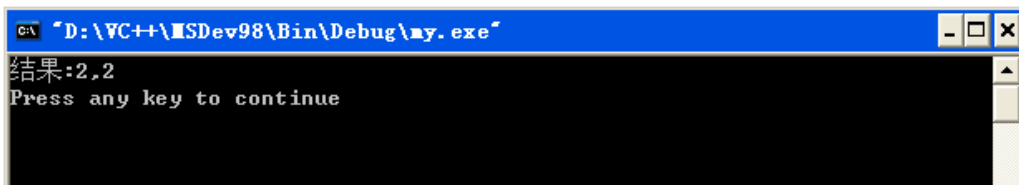


图 10-10 范例 10.11 结果图

【范例 10.12】现假设有若干人的信息要记录，其中包括学生和老师。学生的信息包括姓名、性别、编号、职业、平均成绩，老师的信息包括姓名、性别、编号、职业、任课科目。编写一个程序，从键盘输入数据，并将结果输出至屏幕。

分析：学生与老师要记录的信息只有平均成绩和任课科目不同，因此可以将其定义为共用体类型，其余的用结构体来表示。

范例 10.12 代码实现

```
01     #include <stdio.h>  
02     struct  
03     {  
04         char n[10];  
05         char sex;  
06         char num[20];  
07         char j;  
08         union  
09         {  
10             float score;  
11             char course[10];  
12         }a;  
13     }s[3];                                /*定义了三个结构体嵌套变量*/
```



```

14 void main()
15 {
16     int i;
17     for(i=0;i<3;i++)
18     {
19         scanf("%s %c %s %c",&s[i].n,&s[i].sex,&s[i].num,&s[i].j);
20         if(s[i].j=='s')                /*若 s[i].j 等于字符 s*/
21             scanf("%f",&s[i].a.score);    /*从键盘输入数据保存至 s[i].a.score 中*/
22         else                            /*否则从键盘输入数据至 s[i].a.course 中*/
23             scanf("%s",s[i].a.course);
24     }
25     for(i=0;i<3;i++)
26     {
27         if(s[i].j=='s')                /*输出结构体信息至屏幕*/
28             printf("%s,%c,%s,%c,%f\n",s[i].n,s[i].sex,s[i].num,s[i].j,s[i].a.score);
29         else
30             printf("%s,%c,%s,%c,%s\n",s[i].n,s[i].sex,s[i].num,s[i].j,s[i].a.course);
31     }
32 }

```

【代码分析】本例为结构体与共用体嵌套使用范例，详细代码分析如下：

- 第 17~24 行，实现数据的输入。若 s[i].j 为字符 s 则为学生，输入其平均成绩，用共用体成员 score 来保存，若 s[i].j 为字符 t 则为老师，输入其任课题目，用共用体成员 course 数组来保存。

【运行结果】该程序的执行结果如图 10-11 所示。

图 10-11 范例 10.12 结果图



10.6 枚举型

在日常的生活中，人们经常会碰到一个问题有多个可能的值。例如，一个星期中天数包括星期一、星期二、星期三……星期天，方向控制键有上、下、左、右等。枚举也属于用户构造类型，它是将所有可能的值一一列举出来再来进行选择。枚举类型定义形式如下：

```
enum 枚举类型名{枚举元素表};
```



由浅入深学 C 语言——基础、进阶与必做 430 题

例如：

```
enum color{red,blue,green,white,black};
```

其中 `enum` 为关键字，`color` 为枚举类型名，`red`、`blue`、`green`、`white` 和 `black` 为枚举元素，即所有可能的情况。

定义了枚举类型后，可以用来定义枚举类型的变量。例如：

```
enum color a,b;
```

定义了两个枚举类型变量 `a` 和 `b`，其取值范围在枚举元素表之内，只能为其中的某一个值。如：

```
a=red;
b=white;
```

枚举类型具有以下 4 个特点。

(1) 枚举元素为常量，从起始位置开始值为 0，1，2…。例如：

```
enum color{red,blue,green,white,black};
enum color c;
c=green;
printf("%d",c);
```

上述程序输出结果为 2，因为 `green` 在枚举元素表的第 3 个位置，因此其默认值为 2。

(2) 枚举元素为常量，不能赋值，但可以在定义时进行赋值。例如：

```
enum color{red,blue,green,white,black};
red=5;
green=0;
```

上述程序是错误的，编译时将会出错。

(3) 枚举常量可以用来进行比较。例如：

```
if(c>red) ...
if(c<white) ...
```

因为每个枚举元素都有一个默认值，因此可将枚举类型变量的值与枚举元素进行比较。

(4) 枚举类型不能直接赋值，但可以强制转化进行赋值。例如：

```
c=4;
```

是错误的。

```
c=(enum color)4;
```

将数字强制转化再进行赋值，是正确的。

【范例 10.13】 现有红、黄、蓝、绿、白 5 种颜色，从中取出两个球，打印出所有可能的结果。



分析：球总共有 5 种颜色，分别为红、黄、蓝、绿、白，可以利用枚举类型来求解。

范例 10.13 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      enum color{red,yellow,blue,green,white}; /*自定义枚举类型*/
05      enum color i,j;
06      for(i=red;i<=white;i++)                /*通过嵌套 for 循环输出元素对应的数值*/
07          for(j=red;j<=white;j++)
08              if(i!=j)
09                  printf("%d,%d\n",i,j);
10  }
```

【代码分析】本例为枚举类型应用范例，详细代码分析如下：

- 第 4 行，定义一个枚举类型，其中包括红、黄、蓝、绿、白等 5 种颜色。
- 第 6~9 行，利用 for 循环输出所有可能的结果。

【运行结果】该程序的执行结果如图 10-12 所示。



图 10-12 范例 10.13 结果图



10.7 用户自定义类型

C 语言中可以用关键字 typedef 为已有类型重新定义名称，其定义形式如下：

```
typedef 类型名 标识名;
```

其中类型名可以为 C 语言中的任意类型，标识名为用户自己命名的名字。例如：

```
typedef int integer;
```

上述程序用 integer 表示 int 类型。用 typedef 重新定义类型之后，即可使用该标识符来定



义相应类型的变量，如：

```
integer x;  
int x;
```

上述两种定义变量的方式是等价的，都表示定义一个整型变量 x 。

【范例 10.14】编写一个程序，通过 typedef 自定义类型。

分析：typedef 关键字可以自定义类型。但它不是创建一个新的类型，而是将已有类型用新的标识符来表示。

范例 10.14 代码实现

```
01  #include <stdio.h>  
02  typedef union                                /*通过 typedef 自定义类型*/  
03  {  
04      int a[5];  
05      float b[5];  
06      char c[5];  
07  }ns;  
08  void main()  
09  {  
10      ns x;                                    /*通过自定义类型定义变量*/  
11      printf("共用体所占空间:%d\n",sizeof(x));  
12  }
```

【代码分析】本例用 typedef 自定义类型，详细代码分析如下：

- 第 2~7 行，通过关键字 typedef 为共同体类型定义了一个新的标识符，即 ns。
- 第 10 行，利用新定义的标识符定义变量。

【运行结果】该程序的执行结果如图 10-13 所示。

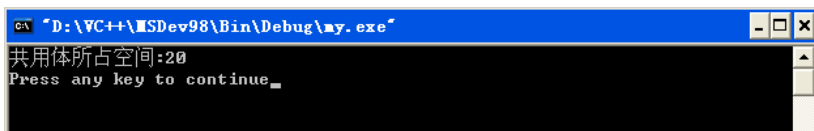


图 10-13 范例 10.14 结果图



10.8 链表

链表是 C 语言中重要的数据结构，常常被用于处理大量数据。虽然数组也可以用来存储大量的数据，但数组必须先声明其大小才能使用。在程序的执行过程中不能改变数组的大小，因此定义时一般将数组定义得很大，这往往会造成存储空间的浪费。

链表是一种动态的存储结构，它可以根据实际情况改变内存空间的大小。若需要则临时开辟存储空间，不需要则释放存储空间，这样可以很好地解决存储空间浪费的问题。链表是一种很复杂的数据结构，它可以分为三种：单向链表、双向链表、循环链表。在本节中将重点介绍



单向链表和双向链表的概念及其应用。

10.8.1 单向链表

单向链表是由若干个结点组成的，每个结点是由数据区域和指针区域两个部分组成。数据区域用来存放用户需要存储的数据，指针区域用来存放下一个结点的地址，便于寻找下一个结点，因此其类型一般为指针类型。

链表中一般都有一个头结点 **head**，用来指向链表的第一个结点，即整个链表的首地址。链表中最后一个结点中指针区域值为空，即为 **NULL**，作为链表的结束标志。数组的存储空间是连续的，而链表是根据实际情况动态申请内存的，因此其存储空间不是连续的。单向链表的示意结构如图 10-14 所示。



图 10-14 单向链表示意图

单向链表每一个结点中的指针区域必须存放下一个结点的地址，才能寻找到下一个结点，否则将不能寻找到下一个结点。单向链表的每一个结点数据结构定义形式如下：

```
struct node
{
    int data;
    struct node *next;
};
```

上述程序定义了一个结点类型 **node**，通过该标识符可以定义变量。其中包含两个部分：整型变量 **data** 和指向 **struct node** 类型的指针 **next**。

【范例 10.15】 通过下面的例子，简单了解链表的概念。

分析：单向链表是链表的一种，它的存储是单向的，其存取必须按顺序进行。每一个结点指针区域应存放下一个结点的地址，只有最后一个结点的指针区域为空。

范例 10.15 代码实现

```
01  #include <stdio.h>
02  struct st
03  {
04      char name[10];
05      int n;
06      float score;
07      struct st *next;          /*结构体类型指针 next*/
08  };
09  void main()
10  {
11      struct st s1={"Wanghu",5324,98.5,NULL};    /*定义结构体变量 s1 并进行初始化*/
12      struct st s2={"Liuli",32121,78.5,NULL};
```



```
13 struct st s3={"Hutao",45221,86.0,NULL};
14 struct st s4={"Yuyan",21243,89.0,NULL};
15 struct st *head; /*定义结构体头指针 head*/
16 head=&s1; /*使 head 指向结构体 s1*/
17 s1.next=&s2; /*使 s1 中 next 指针指向结构体 s2*/
18 s2.next=&s3;
19 s3.next=&s4;
20 s4.next=NULL; /*使 s4 中 next 指针为空*/
21 struct st *p=head; /*定义结构体指针指向链表头结点*/
22 while(p!=NULL) /*通过 while 循环输出所有结点*/
23 {
24     printf("%s,%d,%.1f\n",p->name,p->n,p->score);
25     p=p->next;
26 }
27 }
```

【代码分析】本例为单向链表简单范例，详细代码分析如下：

- 第 2~8 行，定义了一个链表的结点类型，其中包括三种数据及一个指向结构体的指针。
- 第 11~14 行，定义了 4 个结点，并对其进行了初始化。
- 第 15 行，定义一个头结点 head，用来指向第一个结点。
- 第 16~20 行，将第一个结点 s1 的地址赋给头结点 head，后一个结点的地址赋给前一个结点的指针域，最后一个结点的指针赋值为空。
- 第 22~26 行，通过判断结点指针域是否为空来输出每个结点中的数据项，若不为空则输出该结点数据，指向下一个结点，若为空则表示链表结束。

【运行结果】该程序的执行结果如图 10-15 所示。

图 10-15 范例 10.15 结果图

10.8.2 创建及输出链表

在 C 语言中可以使用单向链表动态地存储数据，可以根据实际情况来分配存储空间存储数据，例如下面的例子。

【范例 10.16】通过下面的例子，简单了解链表的应用。

分析：C 语言可以通过 malloc() 函数为链表动态分配存储空间，实现动态存储数据。

范例 10.16 代码实现

```
01 #include <stdio.h>
```



```

02  #include <malloc.h>                                /*包含 malloc.h 头文件*/
03  typedef struct node                                /*自定义结点类型 node*/
04  {
05      int data;
06      struct node *next;
07  }lnode,*link;
08  void create(link *l)                                /*自定义函数 create()*/
09  {
10      int i,n;
11      link p,q=*l;
12      (*l)=(link)malloc(sizeof(node)); /*通过 malloc()函数申请大小为 node 的空间*/
13      (*l)->next=NULL; /*头结点 next 指针为空*/
14      printf("输入链表的大小: ");
15      scanf("%d",&n);
16      for(i=0;i<n;i++)
17      {
18          p=(link)malloc(sizeof(node)); /*动态申请存储空间*/
19          printf("输入数据: ");
20          scanf("%d",&p->data); /*从键盘输入数据保存至 p->data 中*/
21          p->next=NULL; /*结构体变量 p 中指针 next 赋为空*/
22          q->next=p; /*使结点 q 指向结点 p*/
23          q=p; /*将结点 p 赋值给结点 q*/
24      }
25  }
26  void prin(link l)
27  {
28      l=l->next;
29      while(l)
30      {
31          printf("%d ",l->data);
32          l=l->next;
33      }
34  }
35  void main()
36  {
37      link L; /*定义结点类型变量*/
38      create(&L); /*调用 create()函数*/
39      prin(L);
40  }

```

【代码分析】本例创建及输出一个单向链表，详细代码分析如下：

- 第 3~7 行，利用 typedef 定义了结点类型名 node 和结点类型指针 link。
- 第 8~25 行，自定义函数 create()。该函数的功能为创建一个链表，其中 link *l 作为形参接收主函数传来的链表首地址。
- 第 26~34 行，自定义函数 prin()。该函数用来输出链表中每一个结点中的数据，直到最后一个结点为止。



- 第 38~39 行，调用 `create()` 函数创建一个链表，其中 `&L` 作为实参传递给函数。然后利用 `prin()` 函数输出创建的链表中的结点数据。

【运行结果】该程序的执行结果如图 10-16 所示。

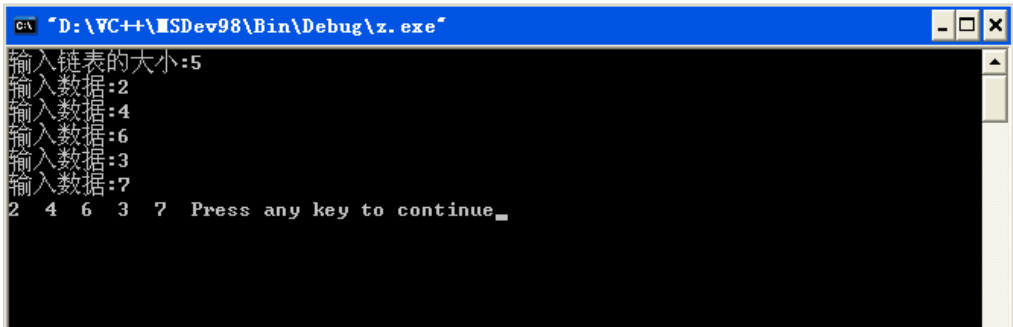


图 10-16 范例 10.16 结果图

10.8.3 双向链表

双向链表也是链表的一种，它包含两个指针。单向链表中只有一个指针，指向下一个结点，因此要在单向链表中寻找一个结点，只能从开始位置逐个往下寻找，这样显得很复杂。而双向链表中包含两个指针，分别指向其前驱和后继，因此可以从该链表中的任一结点出发来寻找某一结点，可以很方便地访问一个结点的前驱和后继。

双向链表与单向链表类似，只是多一个前驱指针而已，其定义形式如下：

```
struct node
{
    int data;
    struct node *pre;
    struct node *next;
};
```

上述程序定义了一个双向链表结点类型，其中包含三个部分：数据域 `data`，指向结点前驱指针 `pre` 和指向结点后继指针 `next`。双向链表的存储结构示意图如图 10-17 所示。



图 10-17 双向链表结构示意图

【范例 10.17】编写一个程序，实现双向链表的输入、输出。

分析：双向链表创建及输出与单向链表类似，但是双向链表比单向链表多一个前驱指针，初始化过程要复杂一些。

范例 10.17 代码实现

```
01  #include <stdio.h>
02  #include <malloc.h>
```



```
03 typedef struct node
04 {
05     int data;
06     struct node *pre;
07     struct node *next;
08 }lnode,*link;
09 void create(link *l)
10 {
11     int i,n;
12     link p,q;
13     (*l)=(link)malloc(sizeof(node));    /*通过 malloc()函数动态申请内存*/
14     (*l)->pre=*l;
15     (*l)->next=NULL;
16     q=*l;                                /*使 q 指向链表头结点*/
17     printf("输入链表的大小: ");
18     scanf("%d",&n);
19     for(i=1;i<=n;i++)
20     {
21         p=(link)malloc(sizeof(node));    /*通过 malloc()函数为变量 p 申请内存*/
22         printf("输入数据: ");
23         scanf("%d",&p->data);            /*从键盘获取数据至 p->data 中*/
24         p->next=NULL;
25         p->pre=q;                        /*使 p 前驱指针 pre 指向 q*/
26         q->next=p;                      /*使 p 后继指针 next 执行 p*/
27         q=p;                            /*将结点 p 赋值给结点 q*/
28     }
29 }
30 void prin(link l)
31 {
32     l=l->next;
33     while(l)
34     {
35         printf("%d ",l->data);
36         l=l->next;
37     }
38 }
39 void main()
40 {
41     link L;
42     create(&L);
43     prin(L);
44 }
```

【代码分析】本题创建及输出一个双向链表，详细代码分析如下：

- 第 3~8 行，定义了双向链表结点类型名 `node` 和类型指针 `link`。
- 第 9~29 行为自定义函数 `create()`，通过该函数来创建双向链表。



由浅入深学 C 语言——基础、进阶与必做 430 题

- 第 33~37 行，通过 while 循环判断下一个结点是否为空来输出链表中的结点。

【运行结果】该程序的执行结果如图 10-18 所示。

```

D:\VC++\MSDev98\Bin\Debug\z.exe
输入链表的大小:5
输入数据:3
输入数据:6
输入数据:8
输入数据:2
输入数据:6
3 6 8 2 6 Press any key to continue_

```

图 10-18 范例 10.17 结果图

10.8.4 链表中插入结点和删除结点

通过链表不仅可以实现数据的输入、输出，而且还可以实现插入、删除等多种操作，实现过程也很简单，例如下面的例子。

【范例 10.18】改写上例，实现链表中结点的插入和删除功能。

分析：要插入一个结点，首先应明确其要插入的位置，可通过改变相邻位置结点指针域的值来实现结点的插入。同样，要删除一个结点，也可以通过改变指针域来实现。

范例 10.18 代码实现

```

01  #include <stdio.h>
02  #include <malloc.h>
03  typedef struct node
04  {
05      int data;
06      struct node *next;
07  }lnode,*link;
08  void create(link *l)
09  {
10      int i,n;
11      link p,q;
12      (*l)=(link)malloc(sizeof(node));
13      (*l)->next=NULL;
14      q=*l;
15      printf("输入链表的大小: ");
16      scanf("%d",&n);
17      for(i=0;i<n;i++)
18      {
19          p=(link)malloc(sizeof(node));
20          printf("输入数据: ");
21          scanf("%d",&p->data);

```



```
22     p->next=NULL;
23     q->next=p;           /*使 q->next 指向 p*/
24     q=p;                 /*将结点 p 赋值给结点 q*/
25 }
26 }
27 void insert(link *l)
28 {
29     int i,n,e;
30     link p=*l,s;          /*使指针 p 指向链表头结点*/
31     printf("输入要插入结点的位置: ");
32     scanf("%d",&n);       /*输入结点的插入位置*/
33     printf("输入要插入结点的值: ");
34     scanf("%d",&e);       /*输入要插入的结点的值*/
35     for(i=1;i<n;i++)
36         p=p->next;        /*将指针移动到要插入的位置*/
37     s=(link)malloc(sizeof(node));
38     s->data=e;             /*将变量 e 赋值给 s->data*/
39     s->next=p->next;       /*使 s->next 指向 p->next*/
40     p->next=s;             /*使 p->next 指向结点 s*/
41 }
42 void ldelete(link *l)
43 {
44     int i,n;
45     link p=*l,s;
46     printf("输入要删除结点的位置: ");
47     scanf("%d",&n);       /*从键盘输入要删除的结点位置*/
48     for(i=1;i<n;i++)      /*将指针移动到要删除结点的位置*/
49         p=p->next;
50     s=p->next;             /*保存要删除结点的下一个结点位置*/
51     p->next=s->next;       /*将该结点前一个结点指针赋值为下一个结点位置*/
52 }
53 void prin(link l)
54 {
55     l=l->next;
56     while(l)
57     {
58         printf("%d ",l->data);
59         l=l->next;
60     }
61     printf("\n");
62 }
63 void main()
64 {
65     link L;
66     create(&L);
67     prin(L);
68     insert(&L);
```



```
69     prin(L);  
70 }
```

【代码分析】本例实现了链表的插入删除功能，详细代码分析如下：

- 第 27~41 行为自定义函数 insert()。该函数的功能为将一个结点插入至链表的指定位置，形成一个新的链表。
- 第 42~62 行，自定义函数 prin()，此函数用来删除链表上特定位置上的结点形成一个新的链表。
- 第 48、49 行，是将指针 p 移至要删除结点的前一个位置。
- 第 50、51 行，使要删除结点的前一个结点指向要删除结点的下一个结点，从而实现删除结点的功能。

【运行结果】该程序的执行结果如图 10-19 所示。

```
C:\> "D:\VC++\MSDev98\Bin\Debug\z.exe"  
输入链表的大小:5  
输入数据:1  
输入数据:2  
输入数据:3  
输入数据:5  
输入数据:6  
1 2 3 5 6  
输入要插入结点的位置:4  
输入要插入结点的值:4  
1 2 3 4 5 6  
输入要删除结点的位置:5  
1 2 3 4 6  
Press any key to continue
```

图 10-19 范例 10.18 结果图



10.9 小结

本章讲解了结构体、共用体、自定义类型及链表等结构。这些结构在 C 语言中经常会使用到，因此读者应认真掌握其概念及应用。其中包括结构体的定义和使用、共用体的定义和使用、自定义类型的定义和使用及链表的定义和使用。



10.10 习题

一、选择题

1. 设有以下结构体

```
struct data  
{  
    int year;  
    int month;
```



```
struct
{
    int day;
    int hour;
    int minute;
}t;
}
struct data s;
```

则下列赋值语句正确的是 ()。

A. day=20;
hour=4;
minute=50;

B. s.day=20;
s.hour=4;
s.minute=50;

C. t.day=20;
t.hour=4;
t.minute=50;

D. s.t.day=20;
s.t.hour=4;
s.t.minute=50

【提示】本题中结构体 data 中嵌套了另外一个结构体，因此引用时必须也使用两个“.”运算符，本题正确选项为 D。

2. 以下枚举类型声明中正确的是 ()。

A. enum color={"red","green","blue"};

B. enum color{"red","green","blue"};

C. enum color{red,green,blue};

D. enum color={red=4,green=1;blue};

【提示】枚举类型元素不需要用引号括起来，定义时枚举元素表紧跟枚举类型之后，中间没有“=”号，可以给元素赋值，例如 red=4 是可以的。

3. 现有如下所示结构体变量，则其所占存储空间为 ()。

```
struct data
{
    int a;
    float b;
    struct
    {
        char c;
        double d;
    }t;
}s;
```

A. 11

B. 13

C. 24

D. 17

【提示】本题中结构体 data 嵌套了结构体 t。int 类型占字节数为 4，float 类型占字节数为 4，char 类型占字节数为 1，double 类型占字节数为 8，因此很多人会误以为结构体 s 所占存储空间为 17，其实这与编译环境有关。不同的编译环境所占字节数可能不同，本书所采用的编译环



由浅入深学 C 语言——基础、进阶与必做 430 题

境为 VC++6.0。data 中嵌套的结构体 t 定义的 char 类型和 double 类型两者在 VC++ 中所占字节空间实际上为 16 个字节，因此结构体 s 所占存储空间为 24 个字节。

4. 下列定义语句中正确的是 ()。

A.

```
struct st
{
    int x,
    int y;
};
```

B.

```
struct st
{
    int x;
    int y;
}
```

C.

```
Struct
{
    int x;
    int y;
}s;
```

D.

```
struct st
{
    int x;
    int y;
}s
```

【提示】本题考察结构体的定义形式，其中定义一个变量后用分号表示语句的结束。结构体大括号之后一定要有分号。

5. 以下程序运行结果为 ()。

```
union my
{
    int x;
    float y;
    char z;
}s;
void main()
{
    s.x=5;
    s.y=6.7;
    s.z='a';
    printf("%c",s.z);
}
```

A. a

B. 值不确定

C. 程序运行出错

D. b

【提示】定义共用体变量 s，对其成员进行了赋值再输出。

6. 以下程序运行结果为 ()。

```
struct node
{
    int data;
    char c;
};
void fun(struct node a)
{
```



```
a.data=4;
a.c='v';
}
void main()
{
    struct node a={5, 'a'};
    fun(a);
    printf("%d,%c",a.data,a.c);
}
```

A. 4, v

B. 5, a

C. 4, a

D. 5, v

【提示】本题调用 fun()函数改变结构体变量 s 的值，但由于其为传值调用，因此变量 s 的值不会发生改变。

7. 以下程序运行结果为 ()。

```
struct node
{
    int data;
    char c;
};
void fun(struct node *a)
{
    a->data=4;
    a->c='v';
}
void main()
{
    struct node a={5, 'a'};
    fun(&a);
    printf("%d,%c",a.data,a.c);
}
```

A. 4, v

B. 5, a

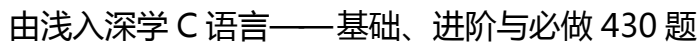
C. 4, a

D. 5, v

【提示】定义结构体变量 a 同时进行了初始化，然后传址方式调用 fun()函数，因此变量 a 的值会发生变化。

8. 以下程序运行结果为 ()。

```
union un
{
    int a;
    char b;
    float c;
}s;
void main()
```

D. 以上都不正确

9. 以下程序运行结果为 ()。

D. 都不正确

10. 以下程序运行结果为 ()。

D. 264

266



二、填空题

1. 现有以下程序，设 `int` 类型占 4 个字节，则其输出结果为_____。

```
void main()
{
    struct st
    {
        char c;
        int x[5];
        float y;
    };
    printf("%d",sizeof(struct st));
}
```

【提示】上述程序通过 `sizeof` 运算符计算 `struct st` 类型占据的存储空间，再进行输出。

2. 若有以下程序，则其运行结果为_____。

```
void main()
{
    union s
    {
        char c[10];
        float x;
        double y;
    };
    printf("%d",sizeof(union s));
}
```

【提示】上述程序定义一个共用体类型 `s`，然后通过 `sizeof` 运算符计算其所占存储空间并输出。

3. 以下程序的运行结果为_____。

```
void main()
{
    union st
    {
        int x;
        char c[10];
    }s;
    s.x=50;
    s.c="abcdef";
    printf("%s",s.c);
}
```

【提示】本题定义共用体变量 `s`，然后对其赋值进行相应的输出。

4. 以下程序的运行结果为_____。



```
void main()
{
    struct st
    {
        int a;
        int b;
        struct
        {
            int x;
        }s;
    }m;
    m.s.x=5;
    m.a=m.s.x-2;
    m.b=m.s.x+3;
    printf("%d,%d",m.a,m.b);
}
```

【提示】本题在结构体变量 `m` 中嵌套了一个结构体变量 `s`，然后对其进行简单的算术操作再输出。

5. 以下程序运行结果为_____。

```
struct node
{
    int num;
    char name[10];
    float score;
    union un
    {
        int i;
        double j;
    }u;
};
void main()
{
    struct node s;
    printf("%d",sizeof(s));
}
```

【提示】本题在结构体中嵌套了共用体变量 `u`，因此其所占存储空间为整型所占存储空间加上浮点型数所占存储空间再加上共用体的存储空间。

6. 以下程序的运行结果为_____。

```
struct node
{
    int data;
    float score;
};
```



```
union un
{
    double d;
    char n[10];
}u;
void main()
{
    printf("%d",sizeof(struct node)+sizeof(u));
}
```

【提示】本题通过 sizeof 运算符计算结构体类型 node 加上共用体变量 u 所占存储空间之和。

7. 以下程序运行结果为_____。

```
void main()
{
    struct st
    {
        char n[10];
        int age;
    };
    struct st s[3]={ "Merra",19, "Wangli",20, "Lili",21};
    printf("%c",s[1].n[0]);
}
```

【提示】本题定义结构体数组 s 并进行初始化，然后输出 s[1].n[0]。

8. 以下程序运行结果为_____。

```
void main()
{
    enum t{a=2,b=4,c,d,e=-1,h};
    printf("%d,%d,%d",c,d,h);
}
```

【提示】本题定义枚举类型 t，然后输出枚举元素对应的值。

9. 以下程序运行结果为_____。

```
void main()
{
    struct node
    {
        int data;
        struct node *next;
    }s[10];
    int i;
    for(i=0;i<10;i++)
        s[i].data=2*i+1;
    for(i=0;i<10;i++)
```



```
printf("%d",s[i].data);  
}
```

【提示】本题定义结构体数组 s，通过 for 循环对其进行赋值，然后输出至屏幕。

10. 以下程序运行结果为_____。

```
void main()  
{  
    union un  
    {  
        double d;  
        char c;  
        int a;  
    }s;  
    printf("%d",sizeof(s));  
}
```

【提示】定义一个共用体变量 s，然后通过 sizeof 运算符计算其存储空间并输出。

11. 以下程序运行结果为_____。

```
void main()  
{  
    enum color{red,white,black,blue};  
    printf("%d,%d,%d,%d",red,white,black,blue);  
}
```

【提示】定义枚举类型 color，输出枚举元素对应的数值。

12. 以下程序运行结果为_____。

```
void main()  
{  
    enum color{red,white,black,blue};  
    enum color c;  
    printf("%d",sizeof(c));  
}
```

【提示】定义枚举类型变量 c，调用 sizeof 运算计算该变量所占存储空间并输出。

13. 以下程序运行结果为_____。

```
struct node  
{  
    char n[10];  
    int score;  
};  
void main()  
{  
    struct node a={"wangli",89},b={"lili",88},c={"yangwu",90};  
    if(strcmp(a.n,b.n)>0) a=b;
```



```
if(strcmp(a.n,c.n)>0) a=c;
printf("%s,%d",a.n,a.score);
}
```

【提示】上述程序比较变量 a, b, c 中的字符串大小, 若 a.n 大于 b.n 则将结构体变量 b 赋值给变量 a, 若 a.n 大于 c.n 则将结构体变量 c 赋值给变量 a。

14. 以下程序运行结果为_____。

```
struct st
{
    int a;
    float b;
}
void main()
{
    struct st s[]={5,89.5,8,91.7};
    struct st *p;
    p=s[0];
    printf("%d,%f\n",p->a,p->b);
    p=s[1];
    printf("%d,%f\n",p->a,p->b);
}
```

【提示】上述程序定义结构体变量 s 并进行初始化, 然后进行相应的输出。

15. 以下程序运行结果为_____。

```
struct node
{
    int data;
    char n[10];
};
void fun(struct node *a)
{
    a->data=10;
}
void main()
{
    struct node s={20, "Mary"};
    printf("%d,%s",s.data,s.n);
    fun(&s);
    printf("%d,%s",s.data,s.n);
}
```

【提示】上述程序中传址调用 fun() 函数, 修改 s.data 的值为 10, 因此结构体变量 s 的值会发生变化。



三、编程题

1. 编写一个程序，用来记录一个学生的信息。

【提示】记录一个学生的信息，可以定义一个结构体用来保存。通过对结构体的操作，实现学生信息的输入和输出。

【核心代码】

```
struct st
{
    char name[10];
    int age;
    char sex;
    int num;
    float score;
}s;
scanf("%s,%d,%c,%d,%f",s.name,&s.age,&s.sex,&s.num,&s.score);
```

2. 现有 5 名学生，要求编写程序从键盘输入这 5 名学生的信息，其中包括学号、姓名、性别、成绩等信息，再将其输出。

【提示】输入 5 名学生的信息，可以定义一个结构体数组来保存，再对结构体数组进行操作实现输入、输出的功能。

【核心代码】

```
struct st
{
    char name[10];
    int age;
    char sex;
    int num;
    float score;
}s[5];
for(i=0;i<5;i++)
scanf("%s,%d,%c,%d,%f",s[i].name,&s[i].age,&s[i].sex,&s[i].num,&s[i].score);
```

3. 编写程序，利用结构体输入若干个学生的信息，将其写至文件中。

【提示】若干个学生的信息用结构体来保存，要将信息写至文件中可以通过 `fwrite()` 函数来实现。该函数用来实现文件数据的写入，其具体使用方法在后面文件的章节会讲解到。

【核心代码】

```
struct st
{
    char name[10];
    int age;
    char sex;
    int num;
    float score;
}s[5];
```



```
FILE *fp;
for(i=0;i<5;i++)
scanf("%s,%d,%c,%d,%f",s[i].name,&s[i].age,&s[i].sex,&s[i].num,&s[i].score);
for(i=0;i<5;i++)
fwrite(s,sizeof(struct st),5,fp);
```

4. 创建一个链表，输入若干个学生的信息，其中包括学号、姓名、成绩，将结果输出至屏幕。

【提示】用链表来存储学生的信息，要先定义其结点类型。每一个结点用来存储单个学生的信息，包括学号、姓名、成绩等信息，再将这些结点链接起来形成一条完整的链表。

【核心代码】

```
struct st
{
    char name[10];
    int age;
    char sex;
    int num;
    float score;
    struct st *next;
}node,*link;
void create(link *l)
{
    (*l)=(link)malloc(sizeof(node));
    (*l)->next=NULL;
    q=*l;
    printf("输入链表的大小: ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        p=(link)malloc(sizeof(node));
        printf("输入数据: ");
        scanf("%s%d%c%d%f",p->name,&p->age,&p->sex,&p->num,&p->score);
        p->next=NULL;
        q->next=p;
        q=p;
    }
}
void prin(link l)
{
    l=l->next;
    while(l)
    {
        printf("%d ",l->data);
        l=l->next;
    }
}
```




5. 编写一个函数，在链表的任意位置上插入一个结点。

【提示】在链表上插入结点，则应使指针先移动到要插入的结点之前，然后将结点插入进去，形成一个新的链表。

【核心代码】

```
void linkinsert(linklist *l,int i,int e)
{
    while(p&& j<i-1)
    {p=p->next; ++j;}
    if(!p || j>i-1) ;
    else
    {
        s=(linklist)malloc(sizeof(lnode));
        s->data=e;
        s->next=p->next;
        p->next=s;
    }
}
```

6. 编写一个程序，打印出 5 名学生的信息至屏幕，要求输出单独编写成一个函数。

【提示】定义相应的结构体数组保存 5 名学生的信息，通过调用自定义函数输出这 5 名学生的信息至屏幕。

【核心代码】

```
struct st
{
    char name[10];
    int age;
    float score;
}s[5];
void prin()
{
    for(i=0;i<5;i++)
        printf("%s %d %f\n",s[i].name,s[i].age,s[i].score);
}
```

第 3 篇 C 语言高级应用

第 11 章 程序的灵魂——算法

C 语言中的程序实际上可以看成是数据结构加上算法。要编写一个程序解决实际问题，首先应表示出要处理的问题模型，然后利用算法对该模型进行操作从而解决问题。算法是程序的灵魂，算法的好坏直接关系到程序的好坏。一个好的程序必定有着一个好的算法，能够方便地解决问题。评价一个算法的好坏，可以通过时间复杂度和空间复杂度进行判断。在本节中，将讲解算法的基本概念及其基本的应用。

本章主要涉及的知识点有：

- 算法；
- 数据结构；
- 时间复杂度；
- 空间复杂度；
- 算法的特性。



11.1 了解算法的必要性

算法不仅在 C 语言中起着很大的作用，而且在科学研究中也至关重要。算法相比其他方法来说，更具体、精确，其具有可行性，可操作性等特性。某一个问题有解，则该问题有相应的算法来解决问题。

计算机是通过数学创造出来的，同时它又是数学的创造者。算法是计算机解决问题的步骤，它与数学有着很大的联系。随着信息化的发展，算法的思想、方法已经融入到人们的生活中，影响着人们平时的生活。

学习算法具有以下三方面的意义：

1. 帮助人们全面理解运算能力

很多时候人们以为运算就是将数据进行加、减、乘、除运算，通过自己熟悉的运算法则来计算得出结果。而实际上，一个问题的解决往往是多种的，不止一种。在计算机中通过不同的算法，都可以解决问题。人们可以比较这些算法的好坏，选择好的算法从而提高运行效率，因此算法可以帮助人们更全面的理解运算能力。

2. 培养人们的思维能力

算法是数学的基本内容，可以有效地培养人们的逻辑思维能力。算法是解决问题的步骤，它具有具体化、程序化、精确性、可行性等特点。通过算法来描述解决问题的方法，能够很有



效地提高人们的逻辑思维能力。

例如一个方程 $ax+b=0$ ；人们求解时都会求得 $x=-b/a$ 。但计算机与人的思想不同，它需要通过算法来求解，因此人们可以编写算法来解决问题。这时候，只要考虑 $x=-b/a$ 显然是不正确的。算法具有准确性，显然当 $a=0$ ， $b=0$ 时， x 的解为全体实数。当 $a=0$ ， $b \neq 0$ 时， x 无解。因此编写算法是一个有条理、有思路的过程，可以培养人们的思维能力。



11.2 求最大值算法

求最大值是指找出某些数据中的最大值。设现有 10 个数据保存在数组中，可以通过设置一个临时变量赋值为第一个元素，然后将该临时变量与后面剩余的元素进行比较。若小于后面的元素，则临时变量赋值为该元素，否则不变。如此所有元素比较完后，临时变量中保存的数据即为所有数据中的最大者。

【范例 11.1】 现有 10 个数据，求出其中的最大者。

分析：设置一个临时变量 t ，将其赋值为第 1 个元素。然后将该临时变量与剩余的元素进行比较，若小于后面的元素，则赋值为后面的元素，否则临时变量 t 值不变，直至所有元素比较完为止。

范例 11.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  void main()
03  {
04      int x[10],t,i;
05      for(i=0;i<10;i++)
06      {
07          printf("输入第%d个数: ",i+1);
08          scanf("%d",&x[i]);
09      }
10      t=x[0];                                          /*将 x[0]赋值给变量 t*/
11      for(i=1;i<10;i++)
12          if(t<x[i])                                  /*若 t 值小于 x[i]*/
13              t=x[i];                                /*将 x[i]赋值给变量 t*/
14      printf("最大的数为%d\n",t);
15  }
```

【代码分析】 本例求数据中的最大值，详细代码分析如下：

- 第 11~13 行，利用 for 循环将变量 t 与数组元素比较，若小于该元素则赋值为该元素，否则不变。

【运行结果】 该程序的执行结果如图 11-1 所示。



```
D:\VC++\MSDev98\Bin\Debug\z.exe
输入第1个数:4
输入第2个数:7
输入第3个数:9
输入第4个数:2
输入第5个数:65
输入第6个数:72
输入第7个数:64
输入第8个数:71
输入第9个数:69
输入第10个数:75
最大的数为75
Press any key to continue_
```

图 11-1 范例 11.1 结果图



11.3 求最小值算法

求最小值算法与求最大值算法恰好相反，它是找出一堆数据中的最小者。该算法可以通过以下方法实现：设置一个临时变量赋值为第一个元素，然后将该临时变量与后面剩余的元素进行比较。若大于后面的元素，则临时变量赋值为该元素，否则不变。如此所有元素比较完后，临时变量中保存的数据即为所有数据中的最小者。

【范例 11.2】 现有 10 个数据，求出其中的最小者。

分析：与求最大值类似，可以设置一个临时变量 t ，将其赋值为第一个元素。然后将该临时变量与剩余的元素进行比较，若大于后面的元素，则赋值为后面的元素，否则临时变量 t 值不变，直至所有元素比较完为止，便可求得最小值。

范例 11.2 代码实现

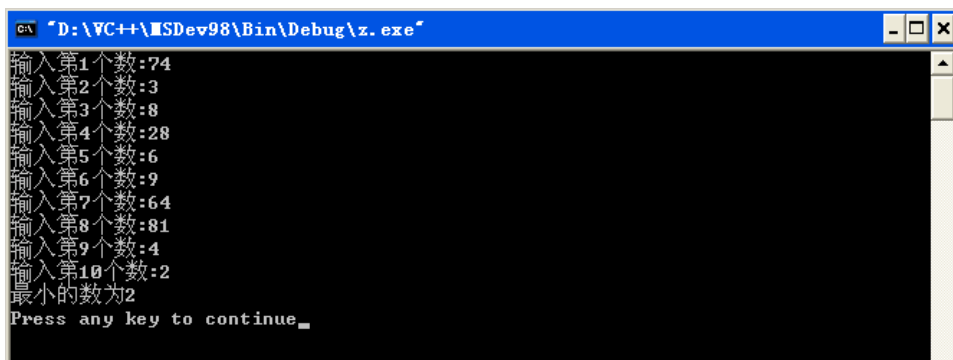
```
01  #include <stdio.h>
02  void main()
03  {
04      int x[10],t,i;
05      for(i=0;i<10;i++)
06      {
07          printf("输入第%d个数: ",i+1);
08          scanf("%d",&x[i]);
09      }
10      t=x[0];
11      for(i=1;i<10;i++)
12          if(t>x[i])                /*若 t 值大于 x[i]*/
13              t=x[i];              /*将 x[i]赋值给变量 t*/
14      printf("最小的数为%d\n",t);
15  }
```

【代码分析】 本例求数据中的最小值，详细代码分析如下：



- 第 10 行，将数组的第一个元素赋给临时变量 t 。
- 第 11~13 行，将临时变量与数组中剩余的元素进行比较。若大于数组中的元素，将该元素赋值给临时变量 t ，否则变量 t 的值不变。

【运行结果】该程序的执行结果如图 11-2 所示。



```
C:\> "D:\VC++\MSDev98\Bin\Debug\z.exe"
输入第1个数:74
输入第2个数:3
输入第3个数:8
输入第4个数:28
输入第5个数:6
输入第6个数:9
输入第7个数:64
输入第8个数:81
输入第9个数:4
输入第10个数:2
最小的数为2
Press any key to continue_
```

图 11-2 范例 11.2 结果图



11.4 排序算法

排序是指将一个数据序列重新组成一个有序的序列。排列可以通过比较关键字来实现，其中包括直接插入排序、折半插入排序、希尔排序、冒泡排序和选择排序等。每种排序方法有着不同的思想及其实现过程，同时不同的排序方法适用于不同的场合。

若要进行排序的序列中有两个相同的元素，排序前后它们的相对位置没有发生变化，则称该算法是稳定的。若排序后其相对位置发生了改变，则称该算法是不稳定的。

在 C 语言中，评价一个算法的好坏，可以通过算法的空间复杂度和时间复杂度来实现。空间复杂度是指算法所需的存储空间，时间复杂度是指执行算法所需的时间。

下面重点讲解这些排序算法的思想及其应用。

11.4.1 直接插入排序

设有 n 个数据序列，直接插入排序认为第一个元素为有序序列，其余元素为无序序列。其思想为从无序序列中逐个选取元素，在有序序列中进行查找比较，若小于该元素则插在元素的前面，否则插在元素的后面。这样有序序列的长度加 1 并且仍然有序，无序序列的长度减 1。如此经过 $n-1$ 次，所有元素便成为一个有序序列，无序序列长度为 0。

直接插入排序是一种稳定的排序算法，适合于要进行排序的序列基本有序或数据较少的情况。

例如，有 10 个元素，分别为 48, 35, 65, 98, 77, 12, 24, 57, 59, 36，其直接插入排序过程如下：

```
初始时：
{48}, {35, 65, 98, 77, 12, 24, 57, 59, 36}
```



第 1 次比较:
 {35,48}, {65,98,77,12,24,57,59,36}
 第 2 次比较:
 {35,48,65}, {98,77,12,24,57,59,36}
 第 3 次比较:
 {35,48,65,98}, {77,12,24,57,59,36}
 第 4 次比较:
 {35,48,65,77,98}, {12,24,57,59,36}
 第 5 次比较:
 {12,35,48,65,77,98}, {24,57,29,36}
 第 6 次比较:
 {12,24,35,48,65,77,98}, {57,29,36}
 第 7 次比较:
 {12,24,35,48,57,65,77,98}, {29,36}
 第 8 次比较:
 {12,24,29,35,48,57,65,77,98}, {36}
 第 9 次比较:
 {12,24,29,35,36,48,57,65,77,98}

【范例 11.3】 现有 10 个数据，编写程序利用直接插入排序法对其进行排序并输出结果。

分析：直接插入排序认为第一个元素有序，再将后面的元素与第一个元素进行比较，将该元素大的元素都后移一位，然后将该元素插入对应的位置中。

范例 11.3 代码实现

```
01  #include <stdio.h>
02  int x[11]={0,48,35,65,98,77,12,24,57,59,36};
03  void paixu() /*自定义函数 paixu()*/
04  {
05      int i,j;
06      for(i=2;i<=10;i++)
07      {
08          if(x[i]<x[i-1]) /*若 x[i] 小于 x[i-1]*/
09          {
10              x[0]=x[i]; /*将 x[i] 赋值给 x[0]*/
11              j=i-1; /*将 i-1 值赋给变量 j*/
12              do{ /*do-while 循环*/
13                  x[j+1]=x[j]; /*将 x[j] 赋值给 x[j+1]*/
14                  j--;
15              }while(x[0]<x[j]); /*若 x[0]<x[j] 则继续执行 do-while 循环*/
16              x[j+1]=x[0]; /*将 x[0] 赋值给 x[j+1]*/
17          }
18      }
19      for(i=1;i<=10;i++)
20          printf("%d ",x[i]);
21      printf("\n");
22  }
```



```
23 void main()  
24 {  
25     paixu();  
26 }
```

【代码分析】本例为直接插入排序应用范例，详细代码分析如下：

- 第 2 行，定义了一个二维数组 `x`，包含 10 个元素，其中第 `x[0]` 用做哨兵，临时保存变量。
- 第 3~22 行，自定义函数 `paixu()`，该函数的功能为对 10 个数进行从小到大排序。其中第 6~18 行，利用 `for` 循环分别将剩余的元素与有序元素进行比较，然后移动相应的元素，再将元素插入序列中。
- 第 25 行，调用自定义函数 `paixu()` 进行排序并输出。

【运行结果】该程序的执行结果如图 11-3 所示。

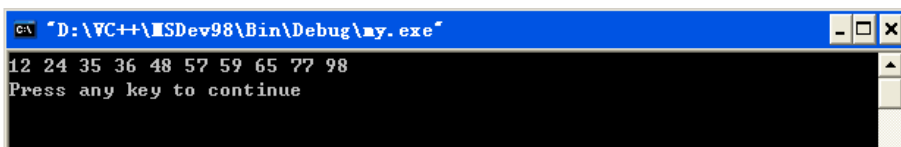


图 11-3 范例 11.3 结果图

11.4.2 折半插入排序

折半插入排序是指在一个有序序列中插入数据形成有序序列。在有序表中进行查找，折半查找是最有效的方法。折半查找设置三个下标 `low`、`high` 及 `mid`，设置 `low` 值为 1，`high` 值为 `n-1`，`mid` 值为 $(low+high)/2$ ，然后将要插入的元素与 `mid` 下标元素进行比较。若小于 `mid` 下标元素，`low=mid+1`，`high` 值不变，若大于 `mid` 下标元素，则 `high=mid-1`，`mid` 的值仍为 $(low+high)/2$ ，再将要插入的元素与 `mid` 下标元素比较，直至找到相应的位置或 `low=high` 为止。

【范例 11.4】现有一个有序序列，从键盘输入一个数据，利用折半查找插入该数据使数列仍然为一个有序序列并输出结果。

分析：折半插入排序是将一个数据插入至有序序列，设置三个变量保存下标，进行比较再在相应的位置插入即可。

范例 11.4 代码实现

```
01 #include<stdio.h>  
02 void Sort(int num[],int n);          /*自定义函数 Sort()声明*/  
03 void main()  
04 {  
05     int n,i;  
06     int num[50];  
07     scanf("%d",&n);                  /*输入元素的个数*/  
08     for(i=1;i<=n;i++)  
09         scanf("%d",&num[i]);  
10     Sort(num,n);                    /*调用 Sort()函数插入排序*/  
11     for(i=1;i<=n;i++)                /*输出插入数据后的序列*/
```



```

12     printf("%d ",num[i]);
13     printf("\n");
14 }
15 void Sort(int num[],int n)                /*自定义函数 Sort()*/
16 {
17     int low,high,i,j,m;
18     for(i=2;i<=n;i++)
19     {
20         num[0]=num[i];                    /*将 num[i] 暂时存放至 num[0]*/
21         low=1;                             /*设置 low 值为 1*/
22         high=i-1;                          /*设置 high 值为 i-1*/
23         while(low<=high)                  /*while 循环*/
24         {
25             m=(low+high)/2;                /*折半*/
26             if(num[0]<num[m])
27                 high=m-1;                  /*插入点在低半区*/
28             else low=m+1;                  /*插入点在高半区*/
29         }
30         for(j=i-1;j>=high+1;j--)
31             num[j+1]=num[j];              /*记录后移*/
32         num[high+1]=num[0];               /*插入数据*/
33     }
34 }

```

【代码分析】本例为折半插入排序应用范例，详细代码分析如下：

- 第 2 行为函数原型声明，因为函数代码在调用之后。
- 第 15~34 行，自定义函数 Sort()。该函数的功能为利用折半查找的思想对数组中的元素进行排序，其中数组作为参数传递给函数。

【运行结果】该程序的执行结果如图 11-4 所示。

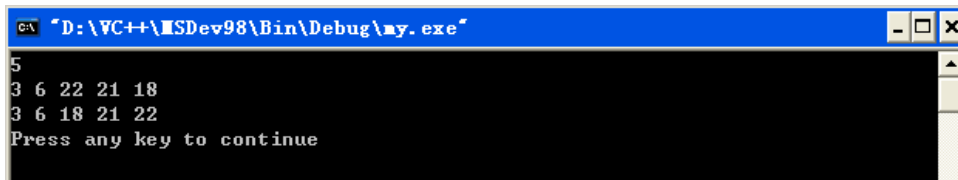


图 11-4 范例 11.4 结果图

11.4.3 希尔排序

希尔排序是一种经过改进后的一种算法，其思想为：将要排序的序列分为若干块，然后对各个块进行排序使每一块中的数据都为有序序列，最后对全体所有元素进行一次插入排序使得所有元素都为有序序列。

希尔排序将要排序的序列分为 d 组，序列中每隔 d 个数据即分为一组。然后对每组的数据进行直接插入排序，再将新的序列按增量 $d1$ 进行分组比较，直至最后增量 d 的值为 1 为止，即全体元素进行比较得到有序序列。



增量 d 一般取 n 值的一半，然后 d 取 d 值的一半，增量的值按规律递减直至最后增量的值为 1 为止。

例如，现有序列{49,39,66,98,78,12,22,56,9,34}，其希尔排序过程如下：

```
第 1 趟分组 d=5:
{49,12},{39,22},{66,56},{98,9},{78,34}
第 1 趟排序:
{12,49},{22,39},{56,66},{9,98},{34,78}
第 2 趟分组 d=3:
{12,39,9,78},{49,56,98},{22,66,34}
第 2 趟排序:
{9,12,39,78},{49,56,98},{22,34,66}
第 3 趟分组 d=1:
{9},{49},{22},{12},{56},{34},{39},{98},{66},{78}
第 3 趟排序:
{9,12,22,34,39,49,66,78,98}
```

【范例 11.5】 从键盘输入 10 个数据，利用希尔算法对其进行排序并输出最终结果。

分析：从键盘输入 10 个数据，可以利用 `scanf()` 函数实现。然后对数组中的元素进行分组比较排序，直至增量的值为 1 为止。

范例 11.5 代码实现

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  void sort(int s[],int n,int d)
04  {
05      int i,j;
06      for(i=d+1;i<n;i++)
07          if(s[i]<s[i-d])                /*若 s[i]小于 s[i-d]*/
08          {
09              s[0]=s[i];                /*将 s[i]赋值给 s[0]*/
10              j=i-d;                    /*将 i-d 值赋给变量 j*/
11              do{                        /*do-while 循环*/
12                  s[j+d]=s[j];
13                  j-=d;                    /*将 j-d 值赋给变量 j*/
14              }while(j>0&& s[0]<s[j]);
15              s[j+d]=s[0];                /*将 s[0]赋给 s[j+d]*/
16          }
17  }
18  void shellsort(int r[],int n)
19  {
20      int d=n;
21      do{
22          d=d/3+1;                        /*将 d/3+1 赋值给 d*/
23          sort(r,n,d);                    /*调用 sort()函数*/
24      }while(d>1);
```



```
25     }
26     void main()
27     {
28         int i,x[11];
29         printf("请输入 10 个整数\n");
30         for(i=1;i<=10;i++)
31             scanf("%d",&x[i]);
32         puts("你输入的序列是: ");
33         for(i=1;i<=10;i++)                /*输出排序前的序列*/
34             printf("%3d",x[i]);
35         shellsort(x,11);
36         printf("\n 希尔排序结果如下: ");
37         for(i=1;i<=10;i++)                /*输出排序后的序列*/
38             printf("%3d",x[i]);
39         printf("\n");
40     }
```

【代码分析】本例利用希尔算法进行排序，详细代码分析如下：

- 第 3~17 行，自定义函数 sort()，该函数的功能为对每一组的数据分别进行排序。
- 第 18~25 行为自定义函数 shellsort()。该函数用来控制增量 d 的变化并且调用 sort() 函数对每一组中的数据排序，直至 d 的值为 1 为止。
- 第 35 行，将 x 数组作为参数传递给 shellsort() 函数对数组中的元素进行排序。

【运行结果】该程序的执行结果如图 11-5 所示。

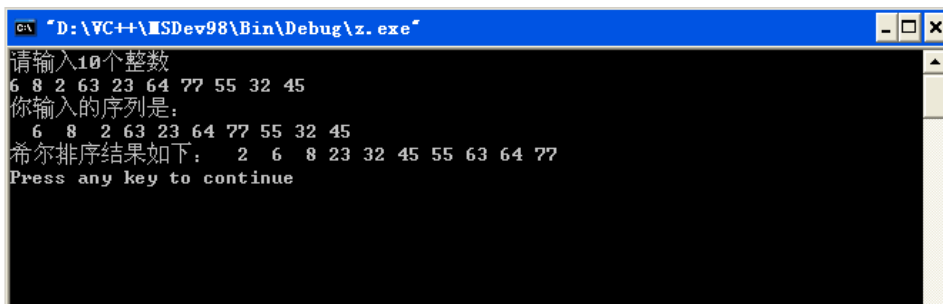


图 11-5 范例 11.5 结果图



技巧：希尔排序是一种不稳定的算法，它需要一个辅助存储空间，用来存放当前正在比较的元素，如本题中将 s[0] 作为辅助的存储空间。

11.4.4 冒泡排序

冒泡排序从无序序列中依次选取最大或最小的关键字置于序列的一端实现排序。

冒泡排序的思想为：将要排序的数据依次进行相邻元素的比较，若为从小到大排列，则将相邻元素的小者放前面，大者放后面，这样值小的元素就会逐步升至元素的起始位置。对待排序数据进行了一次比较，则称为一趟冒泡。第一趟冒泡可将最小值放置待排序数据的起始位置，第二趟排序可将剩余数据中最小的元素放于第二个位置。n 个待排序的元素经过 n-1 排序即可



得到正确的排序序列。

例如，现有序列{-23,34,55,67,-12,32,56,78,-54,87}，则其排序过程如下：

```
第一趟排序后：
-54 34 55 67 -12 32 56 78 -23 87
第二趟排序后：
-54 -23 55 67 -12 32 56 78 34 87
第三趟排序后：
-54 -23 -12 67 55 32 56 78 34 87
第四趟排序后：
-54 -23 -12 32 55 67 56 78 34 87
第五趟排序后：
-54 -23 -12 32 34 67 56 78 55 87
第六趟排序后：
-54 -23 -12 32 34 55 56 78 67 87
第七趟排序后：
-54 -23 -12 32 34 55 56 78 67 87
第八趟排序后：
-54 -23 -12 32 34 55 56 67 78 87
第九趟排序后：
-54 -23 -12 32 34 55 56 67 78 87
```

【范例 11.6】从键盘输入 10 个数，利用冒泡排序将这 10 个数从小到大排列输出至屏幕。

范例 11.6 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[10],i,j,k;
05      printf("input the number\n");
06      for(i=0;i<10;i++)
07          scanf("%d",&x[i]);
08      for(i=0;i<9;i++)                /*冒泡排序过程*/
09          for(j=9;j>i;j--)
10              if(x[i]>x[j])            /*若 x[i]大于 x[j]*/
11              {
12                  k=x[i];              /*通过变量 k 交换 x[i]和 x[j]的值*/
13                  x[i]=x[j];
14                  x[j]=k;
15              }
16      printf("the sort number is: ");
17      for(i=0;i<10;i++)
18          printf("%d ",x[i]);
19      printf("\n");
20  }
```

【代码分析】本例利用冒泡法进行排序，详细代码分析如下：

- 第 8~15 行为冒泡排序，共进行 9 趟比较，第 1 趟比较将最小的数放置于数列的最前



方即 $x[0]$ 中, 第 2 趟比较将剩余数中的最小者放于 $x[1]$ 。依次类推, 9 趟比较后数列即可排好序。

【运行结果】该程序的执行结果如图 11-6 所示。

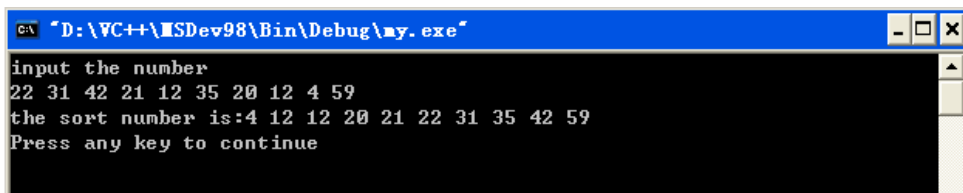


图 11-6 范例 11.6 结果图

11.4.5 选择排序

选择排序法的思想是: 首先设置一个变量保存元素下标, 将该元素与其他元素进行比较。若大于其他元素则交换其下标, 比较完后判断其下标是否发生变化, 若变化则交换两个元素的值, 否则不变。经过一次比较, 即可得出数列中的最小元素并将其放在数列的首位。依此类推, 经过 $n-1$ 次比较即可得到 n 个元素从小到大的序列。

【范例 11.7】从键盘输入 10 个数, 利用选择排序将这 10 个数从小到大排列输出至屏幕。

分析: 选择排序与冒泡排序类似, 但是其设置一个下标, 用来指向序列中较小的元素。

范例 11.7 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int x[8];
05      int i, j, k, temp;
06      for(i=0; i<8; i++)
07          scanf("%d", &x[i]);
08      for(i=0; i<7; i++)                /*选择排序过程*/
09      {
10          k=i;                          /*记录初始下标*/
11          for(j=i+1; j<8; j++)
12              if(x[i]>x[j])              /*若 x[i] 大于 x[j]*/
13                  k=j;                  /*将 j 赋值给 k*/
14          if(k!=i)                       /*若 k 不等于 i*/
15              {temp=x[k]; x[k]=x[i]; x[i]=temp;} /*交换 a[k] 和 x[i] 的值*/
16      }
17      for(i=0; i<8; i++)
18          printf("%4d", x[i]);
19      printf("\n");
20  }
```

【代码分析】本例利用选择排序法对数组进行排序, 详细代码分析如下:



- 第 5 行，定义 4 个整型变量。其中 i 和 j 用于控制 for 循环的执行次数，k 记录数组元素的下标，temp 变量用于元素值的交换。
- 第 8~16 行，用选择排序法对数组进行排序。先用 k 记录每个元素的下标，在将该元素与后面的元素进行比较，若大于后面的元素则改变 k 的值，即记录其下标。最后判断 k 的值是否与 i 的值相等，若不相等则交换这两个元素的值即将该数列中的最小元素放置最前面的位置，因此总共进行 7 次比较即可得到 8 个元素从小到大的排列。

【运行结果】该程序的执行结果如图 11-7 所示。

```
C:\VC++\MSDev98\Bin\Debug\my.exe
5 33 21 22 24 12 15 7
5 7 15 21 22 12 24 33
Press any key to continue
```

图 11-7 范例 11.7 结果图

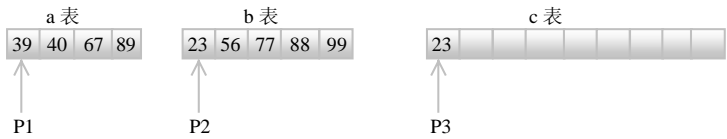
11.4.6 归并排序

归并排序是指将两个或两个以上的有序序列组成一个有序序列。两个有序表合并成一个有序表称为 2-路归并。

2-路归并的思想是设有两个有序表 a 和 b，其长度分别为 a.length 和 b.length。设置两个指针 p1 和 p2 分别指向这两个有序序列 a 和 b，指针 p3 指向序列 c，其长度为 a.length+b.length。比较 p1 和 p2 所指元素的大小，将小者复制到 p3 所指位置，即序列 c 中，然后将 p3 和指向该小者的指针都向后移一位，继续进行比较直至比较完表 a 或表 b 中所有元素为止，最后将表 a 或表 b 中剩余元素复制到表 c 中。

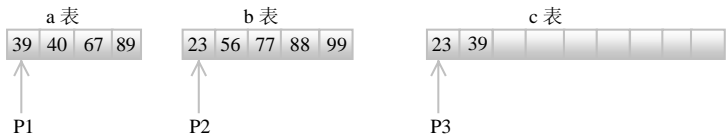
例如，现有两个序列{39,40,67,89}和{23,56,77,88,99}，则其归并排序过程如下：

第 1 趟比较：



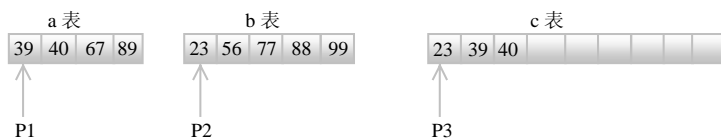
(a) 将较小关键字 23 插入 c 表中

第 2 趟排序：



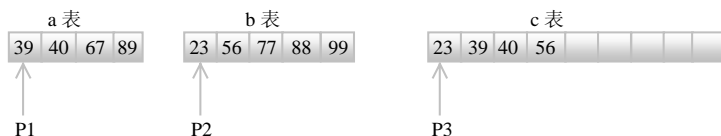
(b) 将较小关键字 39 插入 c 表中

第 3 趟排序：



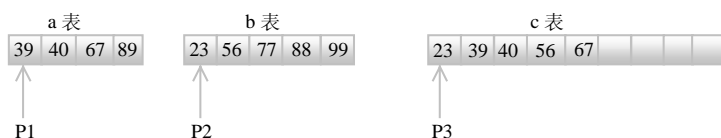
(c) 将较小关键字 40 插入 c 表中

第 4 趟排序:



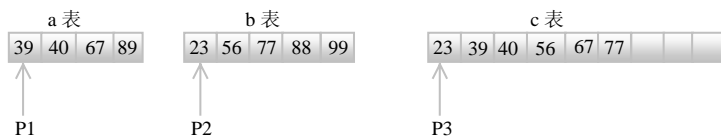
(d) 将较小关键字 56 插入 c 表中

第 5 趟排序:



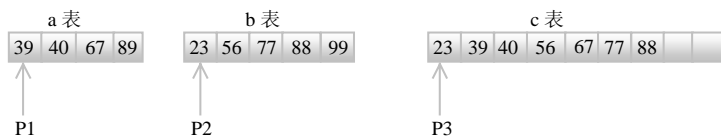
(d) 将较小关键字 67 插入 c 表中

第 6 趟排序:



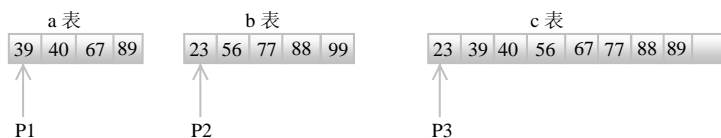
(e) 将较小关键字 77 插入 c 表中

第 7 趟排序:



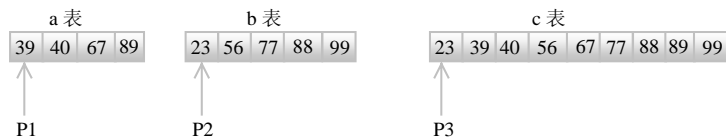
(f) 将较小关键字 88 插入 c 表中

第 8 趟排序:



(g) 将较小关键字 89 插入 c 表中

第 9 趟排序:



(h) 将较小关键字 99 插入 c 表中

【范例 11.8】从键盘输入两个有序序列，利用合并排序将其合并成一个有序序列并输出。

分析：合并排序分别设置两个下标指向两个有序序列，然后比较两个下标所指元素的大小，取出其中的小者，相应的下标加 1，直至将所有元素复制完为止。

范例 11.8 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      int i,n,m,j,k;
05      int a[50],b[50],c[100];
06      printf("输入 a 表的大小: ");
07      scanf("%d",&n);
08      printf("输入 b 表的大小: ");
09      scanf("%d",&m);
10      for(i=0;i<n;i++)
11          scanf("%d",&a[i]);
12      for(i=0;i<m;i++)
13          scanf("%d",&b[i]);
14      i=0;j=0;k=0;
15      while(i<n&&j<m)
16      {
17          if(a[i]>b[j])                /*若 a[i]大于 b[j]*/
18              {c[k]=b[j];j++;k++;}    /*将 b[i]赋值给 c 数组*/
19          else                        /*否则将 a[i]赋值给 c 数组*/
20              {c[k]=a[i];i++;k++;}
21      }
22      while(i<n)                      /*将 a 数组中剩余元素赋值给 c 数组*/
23          {c[k]=a[i];i++;k++;}
24      while(j<m)                      /*将 b 数组中剩余元素赋值给 c 数组*/
25          {c[k]=b[j];j++;k++;}
26      for(i=0;i<n+m;i++)              /*输出 c 数组中所有元素*/
27          printf("%d ",c[i]);
28      printf("\n");
29  }
```

【代码分析】本例利用归并排序对两个有序序列进行合并，详细代码分析如下：

- 第 6~13 行，确定 a 表和 b 表的大小，并从键盘输入数据。其中 n 和 m 为 a 表和 b 表的大小。
- 第 14 行，i, j, k 都赋值为 0，表示将这三个变量分别指向 a 表、b 表、c 表的起始位置。



- 第 15~21 行, 将变量 i 和 j 对应的元素进行比较, 复制其中的小者到 c 表中, 并且使指向小者的下标加 1。
- 第 22~25 行, 将 a 表和 b 表中剩余的元素全部复制给 c 表。

【运行结果】该程序的执行结果如图 11-8 所示。

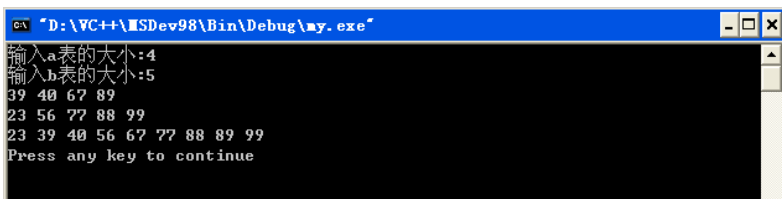


图 11-8 范例 11.8 结果图



11.5 查找算法

在日常生活中, 人们经常需要查找一些数据。当要查找的数据量很大时, 如何快速高效地查找出数据则很重要。若数据都存储在内存中, 称为内部查找, 若数据量很大以至内存存储不下而存放在磁盘上, 则称为外部查找。在 C 语言中, 查找算法包括顺序查找、折半查找、分块查找、哈希表等算法, 在本节中将重点讲解这些查找算法。

11.5.1 顺序查找

顺序查找是指逐个对表中元素进行查询。顺序查找思想: 从表中的开头 (末尾) 开始, 查找整个序列, 将元素与要查找的数值进行比较, 若相等则找到, 否则往下继续查找直至整个序列结束。

【范例 11.9】现有一有序序列, 从键盘输入要查找的数据, 返回其在序列中的位置。

分析: 查找某一特定的数据, 可以用顺序查找的思想, 将该数据与序列中的数据逐个比较进行查找。

范例 11.9 代码实现

```
01  #include <stdio.h>
02  void search(int x[],int n)
03  {
04      int i,flag=0;
05      for(i=0;i<10;i++)
06          if(n==x[i])                /*若 n 等于 x[i]*/
07          {
08              flag=1;                /*flag 值为 1*/
09              printf("查找成功\n");  /*输出查找成功信息*/
10              printf("该数据在序列第%d 个位置\n",i+1); /*输出该元素在序列中的位置*/
11          }
12          if(flag==0)                /*否则输出查找失败信息*/
```




```
13     printf("查找失败\n");
14 }
15 void main()
16 {
17     int x[10]={3,7,12,55,23,5,1,45,2,32},n;
18     printf("输入要查找的数: ");
19     scanf("%d",&n);
20     search(x,n);                /*调用 search()函数查找n*/
21 }
```

【代码分析】本例利用顺序查找查找某个元素，详细代码分析如下：

- 第 2~14 行，自定义函数 search()。该函数的功能为查找某个元素，若存在则返回查找成功信息并输出其在序列中的位置，否则输出查找失败信息。
- 第 20 行，调用 search()函数，其中数组 x 和变量 n 作为参数传递给函数。

【运行结果】该程序的执行结果如图 11-9 所示。

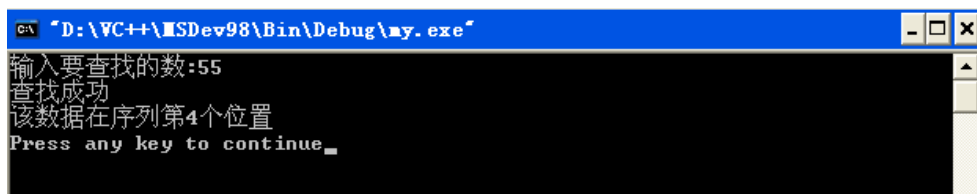


图 11-9 范例 11.9 结果图



注意：顺序查找算法使用起来很简单，但由于其平均查找次数很大，执行效率较低，因此实际上应用不是很多。

11.5.2 折半查找

有序表可以用顺序查找来查找，但用折半查找则比顺序查找效率高很多。

折半查找思想：设置两个下标 low 和 high，分别指向序列的起始位置和末尾。同时另外设置一个下标 mid，其值为 $(low+high)/2$ 。将 mid 所指元素与要查找的元素进行比较，若 mid 所指元素大于要查找的元素，则 $high=mid-1$ ，low 的值不变，即缩小查找空间为左半部分。若 mid 所指元素小于要查找的元素，则 $low=mid+1$ ，high 的值不变，缩小查找空间为右半部分，直至 $low=high$ 值或查找到元素为止。

【范例 11.10】现有一个有序序列，从键盘输入要查找的数，利用折半查找找出该元素在序列中的位置。

分析：折半查找与折半插入排序类似，都是设置下标指向序列，但它是通过折中的方式进行比较查找。

范例 11.10 代码实现

```
01     #include <stdio.h>
02     int zheban(int x[],int n)
```



```

03  {
04      int low=0,high=9,mid;
05      while(low<=high)                /*while 循环,执行条件为 low<=high*/
06      {
07          mid=low+(high-low)/2;        /*将 low+(high-low)/2 赋值给 mid*/
08          if(n>x[mid])                 /*若 n 大于 x[mid]*/
09              low=mid+1;               /*将 mid+1 赋值给 low*/
10          else if(n<x[mid])            /*若 n 小于 x[mid]*/
11              high=mid-1;              /*将 mid-1 赋值给 high*/
12          if(n==x[mid])                /*若 n 等于 x[mid]*/
13              return mid;              /*返回 mid 值*/
14      }
15      return 0;
16  }
17  void main()
18  {
19      int x[10]={4,9,13,23,27,35,44,48,53,58},n;
20      printf("输入要查找的数: ");
21      scanf("%d",&n);
22      int s=zheban(x,n);                /*调用 zheban()函数查找数 n*/
23      if(s==0)                          /*若未找到输出查找失败信息*/
24          printf("查找失败\n");
25      else
26          printf("该元素在序列的第%d 个位置\n",s+1);
27  }

```

【代码分析】本例利用折半查找方法查找元素，详细代码分析如下：

- 第 2~16 行，自定义函数 zheban()。该函数通过折半的方式从有序表中找出相应的元素，并返回该元素的下标。
- 第 22 行，调用 zheban()函数，用变量 s 接收其返回值。
- 第 23~26 行，对 s 的值进行判断。若 s 的值为 0，表示查找失败，若不为 0 则表示查找成功，输出该元素在序列中的位置。

【运行结果】该程序的执行结果如图 11-10 所示。

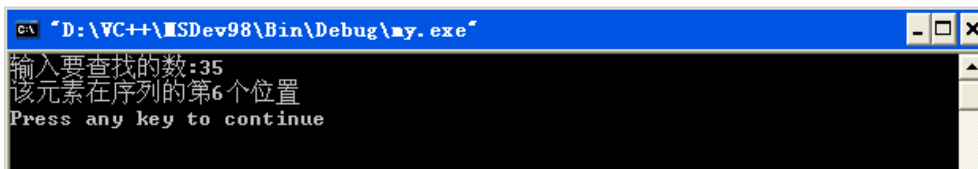
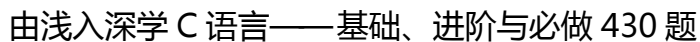


图 11-10 范例 11.10 结果图

11.5.3 分块查找

分块查找是顺序查找的一种改进算法，也称为索引查找。

分块查找思想：将序列分成若干块，其中分块必须有序。分块有序是指后面的块中每一个



例如：有以下序列{22,13,14,7,9,20,34,38,44}，则其可分为如下三组：

$$\{7, 9, 13\}, \{14, 20, 22\}, \{34, 38, 44\}$$

- (1) 性能差于折半查找，高于顺序查找。
- (2) 序列中的每块可以留空余位置，方便数据的插入和删除。
- (3) 分块查索引表的建立需另外的存储空间，时间开销相比折半查找要大些。



11.7 习题

1. 以下程序运行结果为 ()。

【提示】 上述程序调用 `max()` 函数输出 `x` 和 `y` 中的大者。

2. 以下程序运行结果为 ()。

292



```
{
    int i;
    for(i=0;i<n;i++)
        s[i]=s[i+1];
}

void main()
{
    char s[]="avdcagh";
    f(s,4);
    printf("%s",s);
}
```

A. avdcagh

B. vdcagh

C. vdcaagh

D. vdca

【提示】本题通过 f() 函数将数组从 0 到 n-1 的字符都向前移动一位，最后输出字符数组 s 中的内容。

3. 以下程序运行结果为 ()。

```
struct Node
{
    int x;
    struct node *next;
}node[5];

void main()
{
    int i;
    for(i=0;i<5;i++)
        node[i].x=2*i+1;
    for(i=0;i<5;i++)
        printf("%d",node[i].x);
}
```

A. 13579

B. 1357

C. 2468

D. 以上结果都不正确

【提示】将 $2*i+1$ 赋值给 node[i].x，然后通过 for 循环输出至屏幕。

4. 以下程序运行结果为 ()。

```
void fun(int n)
{
    int s=0,i;
    for(i=0;i<n;i++)
        s=s+i*i-1;
    printf("%d",s);
}

void main()
{
```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
int x=5;
fun(5);
}
```

A. 20

B. 21

C. 22

D. 25

【提示】上述程序 fun() 函数变量 s 统计 $i*i-1$ 的累加和，其中 i 从 0 变化至 4，最后输出 s 的值。

5. 现有以下程序，则其运行结果为 ()。

```
void main()
{
    int x;
    char c=39;
    float f=50;
    double y;
    x=f--c*=(y=5.5);
    printf("%d %d %.1f %.1f\n",x,c,f,y);
}
```

A. 92 -42 92.0 5.5

B. 90 -41 92.0 5.5

C. 90 42 92.5 5.5

D. 90 44 92.5 5.5

【提示】上述程序执行 $x=f--c*=(y=5.5)$ 操作后，输出变量 x, c, f, y 的值。

6. 现有以下程序，则其运行结果为 ()。

```
void main()
{
    int x=1,y=2,a=0,b=0,s;
    s=(b=x>y)|| (a=x<y);
    printf("%d,%d,%d\n",a,b,s);
}
```

A. 1, 0, 0

B. 1, 0, 1

C. 0, 0, 1

D. 0, 1, 0

【提示】表达式 $s=(b=x>y)|| (a=x<y)$ 按照从右向左运算，再输出变量 a、b 和 s 的值。

7. 以下程序的运行结果为 ()。

```
void fun(char s[])
{
    int i=0;
    while(s[i]!='\0'&& s[i]!='c')
    {
        printf("%c",s[i]);
        i++;
    }
}
```



```
void main()
{
    char s[]="adccega";
    fun(s);
}
```

A. ad

B. a

C. adega

D. 以上结果都不正确

【提示】上述程序输出字符数组 s 中的内容，直至遇到字符 c 为止。

8. 以下程序运行结果为 ()。

```
void sort(int a[],int n)
{
    int i,j,t;
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
            if(a[i]<a[j])
            {
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
}

void main()
{
    int a[]={1,7,3,8,5};
    sort(a,5);
}
```

A. 13578

B. 87531

C. 17385

D. 以上结果都不正确

【提示】本题通过冒泡排序法对数组 a 中的元素进行排序，然后输出。

9. 以下程序运行结果为 ()。

```
void search(int a[],int n,int m)
{
    int i,flag=0;
    for(i=0;i<n;i++)
        if(a[i]==m)
            flag=1;
    if(flag==1)
        printf("查找成功");
    else
        printf("查找失败");
}
```



```
}  
void main()  
{  
    int a[]={1,2,3,4,5};  
    search(a,3,4);  
}
```

- A. 查找成功 B. 查找失败
C. 运行结果不确定 D. 程序运行出错

【提示】本题通过 search()函数在前 n 个元素中查找元素 m，因此 search(a, 3, 4) 查找会失败。

10. 以下程序运行结果为 ()。

```
void fun(int n)  
{  
    int s=0,i,m=1;  
    for(i=1;i<=n;i++)  
    {  
        s=s+i*m;  
        if(i%2==0)  
            m= -1;  
        else  
            m=1;  
    }  
    printf("%d",s);  
}  
void main()  
{  
    int n=6;  
    fun(6);  
}
```

- A. 5 B. 6
C. -5 D. -4

【提示】本题 fun()函数中 s=s+i*m，若 i 除以 2 余数为 0 则 m 值为-1，否则 m 值为 1。

11. 以下程序运行结果为 ()。

```
void sort(int a[],int n)  
{  
    int i,j,t;  
    for(i=0;i<n-1;i=i+2)  
        for(j=i+1;j<n;j=j+2)  
            if(a[i]<a[j])  
            {  
                t=a[i];  
                a[i]=a[j];  
            }
```



```

    a[j]=t;
}
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
void main()
{
    int a[]={1,7,3,8,5};
    sort(a,5);
}

```

A. 1 3 5 7 8

B. 8 7 5 3 1

C. 1 7 3 8 5

D. 8 1 7 3 5

【提示】本题通过冒泡排序法的思想对数组 a 进行了部分排序，输出数组 a 中的元素至屏幕。

二、填空题

1. 以下程序的运行结果为_____。

```

void main()
{
    int x=50;
    printf("%d\n", (x-30, x>50?x:2*x-1));
}

```

【提示】上述程序计算 (x-30, x>50?x:2*x-1) 的值，然后输出结果至屏幕。

2. 以下程序的运行结果为_____。

```

void main()
{
    char c;
    c='s'-7;
    printf("%d,%c\n",c,c);
}

```

【提示】将字符's'-7 赋值给字符变量 c，然后输出该字符及其对应的 ASCII 码值。

3. 以下程序运行结果为_____。

```

void main()
{
    int x=1,y=3,z=6;
    x=z=10;
    printf("%d,%d,%d",x,y,z);
}

```

【提示】上述程序为简单的赋值语句，进行相应的输出。

4. 以下程序的运行结果为_____。

```

void main()

```




```
{
    int i,s[]={0,0,0,0,0,0};
    for(i=1;i<6;i++)
    {
        s[i]=s[i-1]*2-1;
        printf("%d ",s[i]);
    }
}
```

【提示】上述程序通过 for 循环将 $s[i]$ 赋值为 $s[i-1]*2-1$ ，然后输出 $s[i]$ 。

5. 以下程序若执行时输入 5 sdicdaf，则其输出结果为_____。

```
#include <string.h>
void change(char *s,int n)
{
    char t;
    int i;
    t=s[n-1];
    for(i=n-1;i>0;i=i-1)
        s[i]=s[i-1];
    s[0]=t;
}
void main()
{
    char s[100];
    int n,i,x;
    scanf("%d%s",&n,&s);
    x=strlen(s);
    for(i=1;i<=n;i++)
        change(s,x);
    printf("%s",s);
}
```

【提示】上述程序中从键盘输入数据至变量 n 和 s 中，然后通过 `change()` 函数对数组 s 进行操作再输出结果。

6. 以下程序运行结果为_____。

```
void sort(int a[],int n)
{
    int i,j,t;
    for(i=0;i<n-1;i=i+2)
        for(j=i+1;j<n;j++)
            if(a[i]<a[j])
            {
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
}
```



```
}  
for(i=0;i<n;i++)  
printf("%d ",a[i]);  
}  
void main()  
{  
    int a[]={1,7,3,8,5};  
    sort(a,5);  
}
```

【提示】上述程序定义一个数组 **a**，然后调用 **sort()** 函数利用冒泡排序的思想对数组 **a** 进行排序后输出至屏幕。

7. 以下程序运行结果为_____。

```
void fun(char *s)  
{  
    int i;  
    for(i=0;i<5;i++)  
        printf("%c",s[i]);  
}  
void main()  
{  
    char s[]="abcdefgh";  
    fun(s);  
}
```

【提示】上述程序调用 **fun()** 函数输出字符数组 **s** 中的前 5 个字符。

8. 以下程序运行结果为_____。

```
void fun(int n)  
{  
    int i;  
    float s=0,a=1,b=1;  
    for(i=1;i<=n;i++)  
    {  
        a=a*n;  
        b=b*i;  
        s=s+a/b;  
    }  
    printf("%f",s);  
}  
void main()  
{  
    int n=5;  
    fun(n);  
}
```



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】上述程通过 for 循环计算 $5+5/2+5/3+5/4+1$ 的值。

9. 以下程序运行结果为_____。

```
void main()
{
    int i,a[]={1,5,8,0,10};
    int n=sizeof(a);
    for(i=0;i<n;i++)
        if(a[i]==5)
            printf("查找到数据值为 5 的元素");
}
```

【提示】上述程序通过 for 循环在数组 a 中查找值为 5 的元素。

10. 以下程序运行结果为_____。

```
void f(char *s)
{
    int i;
    for(i=0;i<5;i++)
        s[i]= 'c';
}
void main()
{
    char s[]="abcdefgh";
    f(s);
    printf("%s",s);
}
```

【提示】上述程序中 f()函数将字符数组 s 中的前 5 个字符都置为字符 c，然后输出字符数组中的内容至屏幕。

三、编程题

1. 编写一个程序，实现对字符的加密，然后输出其解密后的字符。

【提示】对字符进行加密，即改变该字符，可以通过移位来实现。

【核心代码】

```
scanf("%c",&c);
c=c^100;
...
c=c^100;
```

2. 编写程序，在屏幕上打印一个杨辉三角的图形。

【提示】杨辉三角图形如下：

```
1
1 1
1 2 1
1 3 3 1
```



1 4 6 4 1

其中每一行第一个数和最后一个数为 1，其他数为上一列相邻的两个元素之和。

【核心代码】

```
printf("1\n");
x[1]=x[2]=1;
printf("%d %d\n",x[1],x[2]);
for(i=3;i<=n;i++)
{
    x[1]=x[i]=1;
    for(j=i-1;j>1;j--)
        x[j]=x[j]+x[j-1];
}
```

3. 编写程序，求两个数的最大公约数。

【提示】设有两个数 x 和 y ，可以利用辗转相除的方法求最大公约数。

【核心代码】

```
scanf("%d%d",&x,&y);
z=x%y;
while(z!=0)
{
    x=y;
    y=z;
    z=x%y;
}
printf("%d",y);
```

4. 一名顾客买价值为 x 元商品，付给售货员 y 元，编写程序实现售货员找钱的张数最小化。

【提示】为了使找钱张数最小化，则应尽可能选取最大金额的人民币找给顾客，由大到小金额人民币依次求解。

【核心代码】

```
int x[]={0,50,20,10,5,2,1};
scanf("%d%d",&m,&n);
z=y-x;
for(i=1;i<7;i++)
{
    a=z/x[i];
    x[i]=a;
    z=z-a*x[i];
}
```

5. 现有 17 个小朋友围成一圈报数，数到 3 的人出列。编写一个程序，求最后的那个小朋友是第几个小朋友。



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】17 个小朋友可以编号为 0~16，然后模拟报数的过程，若报到 3 则为其设置标志，表示已经出了队列，最后输出剩余的编号即可。

【核心代码】

```
for(i=0;i<17;i++)
{
    j=j+x[i];
    if(j==3)
    {
        x[i]=0;
        j=0;
        n=n-1;
    }
}
```

6. 编写程序，实现月份翻译的功能，如输入 4 翻译输出 April。

【提示】输入数字翻译成对应的月份，可以通过 switch 结构来实现，将数值与月份的输出一一对应即可。

【核心代码】

```
switch(month)
{
    case 1: printf("January\n"); break;
    case 2: printf("February"); break;
    case 3: printf("March"); break;
    case 4: printf("April"); break;
    case 5: printf("May"); break;
    case 6: printf("June"); break;
    case 7: printf("July"); break;
    case 8: printf("August"); break;
    case 9: printf("September"); break;
    case 10: printf("October"); break;
    case 11: printf("November"); break;
    case 12: printf("December"); break;
}
```

第 12 章 文 件

文件是 C 语言中的重点，操作起来很复杂。在前面的章节中介绍的程序的输入、输出都是从屏幕进行的，但当数据量很大时，从键盘在屏幕上进行输入、输出则显得很麻烦。因此，在实际编程中，程序员通常将信息保存在文件中，通过对文件进行操作来实现特定的功能。

通过程序将数据写入至文件中，称做文件的输出或对文件进行写操作。若把文件中的内容读出至内存中，称做文件的输入或对文件进行读操作。对文件进行读写操作，可以利用 C 语言提供的库函数来实现，其包含在 `stdio.h` 头文件中。

本章主要涉及的知识点有：

- 文件读操作；
- 文件写操作；
- 文件指针；
- 文件的定位；
- 缓冲文件系统和非缓存文件系统。



12.1 文件简介

C 语言程序中若要输入大量的数据至屏幕，通过键盘输入既复杂，效率又低。这时可以将这些数据先保存在文件中，若需要这些数据，则读取出文件中相应的内容即可。程序执行后的结果也可以保存至文件中，方便以后使用。

在 C 语言中，文件是由一个个字符组成的，文件中的内容称为文件流。文件可以分为两种：ASCII 文件和二进制文件。ASCII 文件是通过字符来存储信息的，其具有可读性。二进制文件是通过二进制形式来存储信息的，即用一连串的二进制数来表示，因此很难读懂。

根据处理方法，文件又可分为缓冲文件和非缓冲文件两种。标准 C 语言中采用的是缓冲文件。

12.1.1 缓冲文件

缓冲文件在内存中会为文件创建一个“缓冲区”，用来作为中介实现文件的读写操作。当进行文件的读操作时，会从文件中读取一部分数据至缓冲区中，再将缓冲区中的内容读取到相应的变量中。当进行文件的写操作时，先将数据写到数据缓冲区中，若缓冲区存储空间满了，再将缓冲区中的内容写至文件中。

缓冲文件实现文件的读、写操作是通过指针来实现的。一般一个文件就要定义一个文件指针，文件指针的定义形式如下：

```
FILE *指针名；
```



例如：

```
FILE *fp;
```

定义了一个名为 `fp` 的文件指针，通过该指针可以进行文件的读、写等操作。

12.1.2 非缓冲文件

非缓冲文件中系统不会创建缓冲区来存储数据，其占用的为操作系统缓冲区，不是用户缓冲区。

缓冲文件是通过文件指针来实现对文件的读和写操作的，而非缓冲文件是通过操作系统提供的功能来实现文件的读和写操作，为低级的输入输出。它只能对二进制文件读和写，但其速度很快，执行效率高，一般使用较少。

12.1.3 文件指针和位置指针

在对一个文件进行操作之前，首先要定义一个文件指针。指针定义之后通过 `fopen()` 函数给文件指针赋值，即使文件指针指向相应的文件。文件指针赋值之后，就可以通过文件指针对文件进行读写等操作。

位置指针用来指明文件当前操作的位置。当调用 `fopen()` 函数打开相应文件之后，文件位置指针指向文件开头的第 1 个字符。当文件位置指针指向文件最后一个字符时，则表示文件结束标志。C 语言还提供了 `feof()` 函数专门用来检测文件位置指针是否移动到末尾。当对文件进行读、写操作时，每次读取或写一个字符后，位置指针向后移动一位，直到移动到文件最后一个字符为止。



12.2 与文件有关的库函数

C 语言中提供了丰富的库函数给用户直接使用，通过这些库函数可以实现文件读写等操作。在本节中将重点介绍这些库函数及其应用。

12.2.1 文件的打开和关闭函数

打开一个文件是通过 `fopen()` 函数来实现的。`fopen()` 函数的调用形式如下：

```
fp=fopen(文件名, 文件操作方式);
```

例如：

```
FILE *fp;  
fp=fopen("a.txt", "w");
```

定义了一个 `FILE` 类型的文件指针，调用 `fopen()` 函数打开当前目录下的 `a.txt` 文本文件，并只能对文件进行写操作，不能进行读操作。文件指针 `fp` 指向了文件 `a.txt`，并指向该文件的起

始位置。



注意：`fp=fopen("a.txt","w");`表示只对当前目录下 `a.txt` 进行写操作，不能对其他目录下的文件进行操作。若要对其他目录下的文件进行操作，则应将绝对路径写至 `fopen()` 函数中。如要调用 `c:\data` 目录下的 `w.txt` 文件，则其调用形式如下：

```
fp=fopen("c:\\data\\w.txt","r");
```

在 C 语言中用 “\\” 表示单斜杠 “\”。

为了判断文件是否打开，可以通过 `fopen()` 的返回值进行判断。例如：

```
if((fp=fopen("a.txt","w"))==NULL)
{
    printf("文件打开错误\n");
    exit(1);
}
```

若文件打开失败，则函数 `fopen()` 会返回一个 `NULL` 值，即为 0 值。因此可以通过判断 `fopen()` 函数的返回值来判断文件是否正常打开。

C 语言中对文件操作提供了多种方式，如表 12-1 所示。

表 12-1 文件操作方式

操作方式	含 义	作 用
“r”	只读方式	打开一个文件进行读操作
“w”	只写方式	打开一个文件进行写操作
“a”	追加方式	打开一个文件进行追加写操作
“rb”	只读方式	打开一个二进制文件进行读操作
“wb”	只写方式	打开一个二进制文件进行写操作
“ab”	追加方式	打开一个二进制文件进行追加写操作
“r+”	读写方式	打开一个文件进行读写操作
“w+”	读写方式	创建一个文件进行读写操作
“a+”	读写方式	打开一个文件进行读写操作
“rb+”	读写方式	打开一个二进制文件进行读写操作
“wb+”	读写方式	创建一个文件进行读写操作
“ab+”	读写方式	打开一个二进制文件进行读写操作



注意：

- (1) 若用 “r” 方式打开一个文件，表示对文件进行读操作而且该文件必须已经存在，不能打开不存在的文件，否则会出错。
- (2) 若用 “w” 方式打开一个文件，表示对文件写操作。若该文件不存在，则会自动创建该名字的文件。若文件已存在，则在打开时会将文件中的内容删除，再对文件进行写入数据。
- (3) 若用 “a” 方式打开一个文件，表示对文件追加写，即将数据写在文件末尾之后。文件必须存在，否则会出错。



(4) 若用“r+”、“w+”、“a+”方式打开文件，表示对文件进行读写操作。用“r+”打开文件时，文件必须已经存在。用“w+”打开文件时，会创建一个文件，然后写入数据，可以进行读写操作。用“a+”打开文件时，指针会移至文件的末尾，然后向文件中添加数据。

(5) 程序运行过程中，系统会自动打开相应的标准文件。因此从终端（键盘、鼠标等）输入、输出时，不需要打开标准文件。

当对一个文件的操作结束以后，为防止数据内容被改变，就应关闭文件。关闭文件之后，文件指针不再指向文件，除非重新初始化。

文件的关闭通过调用 `fclose()` 函数实现，其调用形式如下：

```
fclose(文件指针名);
```

例如：

```
FILE *fp;  
fclose(fp);
```

通过 `fclose()` 函数切断了指针 `fp` 与文件之间的联系，即关闭了文件。

`fclose()` 函数也有一个返回值，若文件正常关闭则其返回值为 0，若文件无法正常关闭则其返回值为 1，可以通过判断 `fopen()` 函数的返回值来判断文件是否被正常关闭。

12.2.2 文件的读写函数

文件读写函数有很多种，其中包括 `fputc()` 函数、`fgetc()` 函数、`fread()` 函数、`fwrite()` 函数、`fprintf()` 函数和 `fscanf()` 函数等。

1. `fputc()` 函数

调用形式：

```
fputc(c, fp);
```

功能：将字符 `c` 中存储的字符写至 `fp` 所指的文件中。

其中 `c` 为要写入的字符，它可为字符常量也可以为字符变量，`fp` 指向要写入的文件。`fputc` 也有一个返回值，若写入成功则返回写入的字符，若写入失败返回 EOF，其值为 -1。

在前面的章节中讲解过 `putchar()` 函数，其实该函数是由 `fputc()` 函数派生而来的。

【范例 12.1】 编写一个程序，从键盘输入一个字符写至文件中。

分析：将一个字符写至文件中，可以先定义一个文件指针，指向要写入的文件。再通过 `fputc()` 函数写入字符至指针指向的文件中。

范例 12.1 代码实现

```
01  #include <stdio.h>                                /*包含 stdio.h 头文件*/  
02  void main()  
03  {  
04      FILE *fp;                                       /*定义文件指针 fp*/
```



```
05  char c;
06  printf("输入要写入的字符:");
07  scanf("%c",&c);
08  if((fp=fopen("file.txt","w"))==NULL) /*若 fopen() 函数打开文件出错则输出信息*/
09  {
10      printf("打开文件出错\n");
11      exit(1);
12  }
13  fputc(c,fp);                          /*将字符变量 c 写至 fp 所指文件中*/
14  fclose(fp);                          /*调用 fclose() 函数关闭文件指针 fp*/
15  }
```

【代码分析】本例为 fputc() 函数的简单范例，详细代码分析如下：

- 第 4 行，定义了一个文件指针 fp。
- 第 8~12 行，通过判断 fopen() 函数的返回值确定 file.txt 是否被正常打开。若打开文件出错，则输出“打开文件出错”的信息。
- 第 13 行，将输入的字符写至 fp 所指文件中。
- 第 15 行，利用 fclose() 函数关闭文件指针。

【运行结果】键盘输入：C

file.txt 文本中内容：C

【范例 12.2】利用 fputc() 函数将一串字符串写入文件中，直到遇到“*”截止。

分析：写入一串字符到文件中，可以利用 while 循环和 fputc() 函数来实现。while 循环对每一个字符进行判断，若不为“*”则写入文件中，若为“*”则退出 while 循环，停止文件的写操作。

范例 12.2 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      FILE *fp;
05      char c;
06      if((fp=fopen("file.txt","w"))==NULL)
07      {
08          printf("打开文件出错\n");
09          exit(1);
10      }
11      c=getchar();
12      while(c!='*')                          /*while 循环，执行条件为 c 不等于字符* */
13      {
14          fputc(c,fp);
15          c=getchar();                        /*重新从键盘获取字符*/
16      }
17      fclose(fp);
```



```
18 }
```

【代码分析】本例利用 `fputc()` 函数写入字符串，详细代码分析如下：

- 第 12~16 行为 `while` 循环。先输入一个字符，然后在 `while` 循环中对字符进行判断，若不为 `*` 字符，则写至文件中并且获取下一个字符进行判断，直到遇到字符 `*` 退出 `while` 循环。

【运行结果】输入一串字符：HELLO C!

file.txt 文本中内容:HELLO C!

2. `fgetc()` 函数

调用形式：

```
c=fgetc(fp);
```

功能：从某一文件中读取一个字符，同时文件打开方式必须为读方式或读写方式。例如：

```
FILE *fp;
fp=fopen("file.txt", "r");
c=fgetc(fp);
```

表示从 `file.txt` 文本文件中读取一个字符，因为位置指针没有赋值，因此读取的是该文件中的第一个字符。若 `fgetc()` 函数读到文件的末尾，则其会返回一个结束符 `EOF`，其值为 `-1`。因此如果想要读取文件中的所有内容，上述程序可改为如下形式：

```
FILE *fp;
fp=fopen("file.txt", "r");
c=fgetc(fp);
while(c!=EOF)
{
    putchar(c);
    c=fgetc(fp);
}
```

【范例 12.3】现磁盘上有一名为 `cfile.txt` 的文本文件，编写程序读出该文件中的所有内容。

分析：读取一个文件中所有内容，可以利用 `fgetc()` 函数读取一个字符，然后对读取出的字符进行判断。若为 `EOF` 则表示文件结束，退出 `while` 循环停止文件的读操作。若不为 `EOF`，则继续读取文件中的下一个字符并输出。

范例 12.3 代码实现

```
01 #include <stdio.h>
02 void main()
03 {
04     FILE *fp;
05     char c;
06     if((fp=fopen("cfile.txt", "r"))==NULL) /*以读的方式打开 cfile.txt 文件*/
```



```
07  {
08      printf("打开文件出错\n");
09      exit(1);
10  }
11  c=fgetc(fp);                /*利用 fgetc()函数从 fp 所指文件中获取字符*/
12  while(c!=EOF)               /*while 循环, 执行条件为 c!=EOF*/
13  {
14      putchar(c);             /*调用 putchar()函数输出字符变量 c*/
15      c=fgetc(fp);           /*从 fp 所指文件中获取字符至变量 c 中*/
16  }
17  fclose(fp);
18  }
```

【代码分析】本例利用 `fgetc()` 函数读取文件中的内容, 详细代码分析如下:

- 第 6~10 行, 以读的方式打开 `cfile.txt` 文本文件并判断是否正常打开文件。
- 第 12~16 行, 通过判断字符是否为 `EOF`, 即值为 -1, 来确定文件指针是否移动到文件末尾, 从而输出文件中的所有内容。

【运行结果】`file.txt` 文本中的内容: my C Program!

输出内容: my C Program!

3. `fread()` 函数

调用形式:

```
fread(buff,size,count,fp);
```

功能: 从文件中读取一个数据块。其中 `buff` 为指针类型, 表示从文件中读取数据存放的地址。`size` 表示一次读取的字节数, `count` 表示进行多少次的读操作, `fp` 为文件指针, 指向要进行操作的文件。

例如若有以下结构体:

```
struct st
{
    char name[20];
    int age;
    char sex;
    float score;
}s[10];
```

定义了一个结构体数组 `s`, 其中包含 10 个元素, 每一个元素都包含姓名、年龄、性别、分数等信息数据。

若这 10 名学生的信息都已存储在文件中, 要从文件中读取出这 10 名学生的信息, 则可以用如下所示程序实现:

```
for(i=0;i<10;i++)
    fread(&s[i],sizeof(struct st),1,fp);
```



执行完上述语句，结构体数组 `s` 分别保存了每一个学生的信息，其中 `&s[i]` 表示第 `i` 个元素的地址，因为 `fread()` 函数是将数据写至变量地址中的，因此该参数必须为变量的地址。

【范例 12.4】 现有一文件 `data.txt` 中存储了 10 名学生的信息，编写程序读取出文件中的内容，并将其显示在屏幕上。

分析：要读取文件中的学生的信息，可以通过 `fread()` 函数实现。`fread()` 函数可以对某一块数据进行读取，因此可以每次读取出一个学生的信息，将其保存在相应的结构体变量中。最后将结构体变量中的成员输出，即为每一个学生的信息。

范例 12.4 代码实现

```
01  #include <stdio.h>
02  struct st
03  {
04      char name[20];
05      int age;
06      char sex;
07      float score;
08  }s[10];
09  void main()
10  {
11      FILE *fp;
12      int i;
13      if((fp=fopen("data.txt","r"))==NULL) /*以读的方式打开 data.txt 文本文件*/
14      {
15          printf("打开文件出错\n");
16          exit(1);
17      }
18      for(i=0;i<10;i++) /*从 fp 所指文件中读取数据至变量 s[i]中*/
19          fread(&s[i],sizeof(struct st),1,fp);
20      for(i=0;i<10;i++) /*通过 for 循环输出每名学生的信息*/
21          printf("%s,%d,%c,%f\n",s[i].name,s[i].age,s[i].sex,s[i].score);
22      fclose(fp);
23  }
```

【代码分析】 本例利用 `fread()` 函数读取文件中的内容，详细代码分析如下：

- 第 2~8 行，定义了一个结构体数组 `s`，用来保存学生的信息。
- 第 18、19 行，通过 `fread()` 函数读取文件中的内容并保存至相应的结构体变量中。
- 第 20、21 行，输出每一个结构体元素的值，即文本文件中学生的信息。

【运行结果】 `data.txt` 文本中内容如下：

```
zhangho 20 w 89.5
wangli 19 w 80.5
wangdo 21 m 87.5
lili 22 w 90.0
liuho 21 m 78.5
```



```
dinglile 23 m 79.0
xiaofan 22 m 79.5
caini 21 w 84.0
xiudo 22 m 85.5
xueming 23 m 88.0
```

输出结果如下所示。

```
zhangho,20,w,89.5
wangli,19,w,80.5
wangdo,21,m,87.5
lili,22,w,90.0
liuho,21,m,78.5
dinglile,23,m,79.0
xiaofan,22,m,79.5
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```

4. fwrite()函数

调用形式:

```
fwrite(buff,size,count,fp);
```

功能: 向文件中写入一个数据块。与 fread()函数类似, 其中 buff 为指针类型, 表示向文件中写入数据的存储地址。size 表示一次写入的字节数, count 表示进行多少次写操作, fp 为文件指针, 指向要进行操作的文件。

例如同样若有以下结构体:

```
struct st
{
    char name[20];
    int age;
    char sex;
    float score;
}s[10];
```

设上述结构体已初始化, 要将这 10 名学生的信息写入文件中, 则可以通过以下所示程序实现:

```
for(i=0;i<40;i++)
    fwrite(&s[i],sizeof(struct st),1,fp);
```

【范例 12.5】 改写范例 12.4, 实现从键盘输入这 10 名学生的信息, 将这些信息写入文件中。

分析: 要写入学生的信息至文件中, 可以利用 fwrite()函数来实现。fwrite()函数每次写入一个学生的信息至文件中, 进行 10 次调用则可将 10 名学生的信息写入文件中。因为是要向文件中写入数据, 因此文件还必须以写的方式打开。



范例 12.5 代码实现

```
01  #include <stdio.h>
02  struct st
03  {
04      char name[20];
05      int age;
06      char sex;
07      float score;
08  }s[10];
09  void main()
10  {
11      FILE *fp;
12      int i;
13      if((fp=fopen("student.txt","w"))==NULL)    /*以写的方式打开 fp 所指文件*/
14      {
15          printf("打开文件出错\n");
16          exit(1);
17      }
18      for(i=0;i<10;i++)
19          scanf("%s%d%c%f",s[i].name,&s[i].age,&s[i].sex,&s[i].score);
20      for(i=0;i<10;i++)                          /*将结构体信息写至 fp 所指文件中*/
21          fwrite(&s[i],sizeof(struct st),1,fp);
22      fclose(fp);
23  }
```

【代码分析】本例利用 `fwrite()` 函数向文件中写入数据，详细代码分析如下：

- 第 18、19 行，利用 `for` 循环和 `scanf()` 函数，将从键盘输入的数据保存至相应的结构体变量中。
- 第 20、21 行，通过 `for` 循环将 10 个结构体变量写入 `fp` 所指的 `student.txt` 文本文件中。

【运行结果】从键盘输入如下内容：

```
zhangho,20,w,89.5
wangli,19,w,80.5
wangdo,21,m,87.5
lili,22,w,90.0
liuho,21,m,78.5
dinglile,23,m,79.0
xiaofan,22,m,79.5
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```

`student.txt` 文本中的内容如下所示。

```
zhangho,20,w,89.5
wangli,19,w,80.5
wangdo,21,m,87.5
```



```
lili,22,w,90.0
liuho,21,m,78.5
dinglile,23,m,79.0
xiaofan,22,m,79.5
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```

5. fprintf()函数

调用形式:

```
fprintf(文件指针, 格式, 输出变量名);
```

功能: 对文件进行格式化写。例如:

```
fprintf(fp, "%d,%c", n, c);
```

其作用为将整型变量 `n` 和字符变量 `c` 按 `%d,%c` 的格式写至指针 `fp` 所指的文件中。

若 `n=5`, `c='x'`;则 `fp` 所指文件中的内容为:

```
5,x
```

`fprintf()`函数使用起来很方便,也很好理解。但 `fprintf()`函数在输入时要将 ASCII 码转化为相应的二进制数。因其执行效率较低,因此很少使用,一般用 `fread()`函数替代它。

【范例 12.6】 改写范例 12.5, 利用 `fprintf()`函数, 写入若干个学生的信息至文件中。

分析: `fprintf()`函数对文件进行写操作时,要写入多少个变量,则应有多少个格式,即 `fprintf()`函数参数中的格式一定要与输出变量一一对应。

范例 12.6 代码实现

```
01  #include <stdio.h>
02  struct st
03  {
04      char name[20];
05      int age;
06      char sex;
07      float score;
08  }s[5];
09  void main()
10  {
11      FILE *fp;
12      int i;
13      if((fp=fopen("stud.txt","w"))==NULL) /*以写的方式打开 stud.txt 文本文件*/
14      {
15          printf("打开文件出错\n");
16          exit(1);
17      }
18      for(i=0;i<5;i++)
```




```
19 scanf("%s%d%c%f",s[i].name,&s[i].age,&s[i].sex,&s[i].score);
20 for(i=0;i<5;i++) /*利用 fprintf()函数将 s[i]写入至相应文件中*/
21 fprintf(fp,"%s,%d,%c,%f",s[i].name,s[i].age,s[i].sex,s[i].score);
22 fclose(fp);
23 }
```

【代码分析】本例利用 `fprintf()` 函数向文件中写入数据，详细代码分析如下：

- 第 20、21 行，通过 `fprintf()` 函数使结构体按指定的格式写入至 `fp` 所指的文件中。

【运行结果】从键盘输入如下内容：

```
zhangho 20 w 89.5
wangli 19 w 80.5
wangdo 21 m 87.5
lili 22 w 90.0
liuho 21 m 78.5
```

`stud.txt` 文本文件中的内容：

```
zhangho,20,w,89.5
wangli,19,w,80.5
wangdo,21,m,87.5
lili,22,w,90.0
liuho,21,m,78.5
```

6. `fscanf()` 函数

调用形式：

```
fscanf(文件指针, 格式, 输入变量名);
```

功能：对文件进行格式化读。例如：

```
fprintf(fp,"%d,%c",&n,&c);
```

若文本中内容为 `5,c`，则 `n` 赋值为 `5`，`c` 赋值为字符 `c`。

`fscanf()` 函数使用起来也很简单，但其在输出时要将二进制数转换为字符形式。因其执行效率也很低，因此一般用 `fwrite()` 函数来替换它。

【范例 12.7】改写范例 12.6，从 `data.txt` 文件中读取信息至屏幕。

分析：`fscanf()` 函数是将文本文件中的内容读取至相应的变量中，其中格式参数的个数也要与输入变量对应。

范例 12.7 代码实现

```
01 #include <stdio.h>
02 struct st
03 {
04     char name[20];
05     int age;
06     char sex;
```



```
07     float score;
08     }s[5];
09     void main()
10     {
11         FILE *fp;
12         int i;
13         if((fp=fopen("stud.txt","w"))==NULL)
14         {
15             printf("打开文件出错\n");
16             exit(1);
17         }
18         for(i=0;i<5;i++)                /*从文件中获取信息保存至结构体 s[i]中*/
19             fscanf(fp,"%s,%d,%c,%f",s[i].name,&s[i].age,&s[i].sex,&s[i].score);
20         for(i=0;i<5;i++)                /*输出每名学生的信息*/
21             printf("%s,%d,%c,%f\n",s[i].name,s[i].age,s[i].sex,s[i].score);
22         fclose(fp);
23     }
```

【代码分析】本例利用 `fscanf()` 函数从文件中读取数据，详细代码分析如下：

- 第 18、19 行，通过 `fscanf()` 函数从 `fp` 所指文件中读取数据至变量 `s[i]` 中。

【运行结果】`stud.txt` 文本文件中内容如下：

```
dinglile,23,m,79.0
xiaofan,22,m,79.5
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```

输出结果：

```
dinglile,23,m,79.0
xiaofan,22,m,79.5
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```



12.3 文件定位函数

每一个文件都有一个位置指针，用于指向当前进行读写的位置。若对文件进行读写操作，每次读写完一个字符，则位置指针会向后移动一个字符，即移动到下一个位置。若想对文件中的其他位置进行读写，可以利用 C 语言提供的库函数来改变位置指针的位置。

12.3.1 `feof()` 函数

调用形式：



```
feof(文件指针);
```

功能：判断文件指针是否到文件末尾。例如：

```
FILE *fp;  
feof(fp);
```

判断指针 `fp` 是否移动到文件的末尾，若文件指针移动到末尾，则 `feof()` 函数的返回值为 1，否则该函数的返回值为 0。

12.3.2 rewind()函数

调用形式：

```
rewind(文件指针);
```

功能：使位置指针重新移动到文件的开头。例如：

```
rewind(fp);
```

通过调用 `rewind()` 函数，使指针 `fp` 指向文件的起始位置。

【范例 12.8】从磁盘中读取一文件显示在屏幕上，然后将该文件中内容复制到另一文件中。

分析：从磁盘中读取文件，可定义一个文件指针指向该文件。通过文件指针和文件操作函数读取文件中的内容至屏幕上，然后将文件中的内容复制到另一文件中。

范例 12.8 代码实现

```
01  #include <stdio.h>  
02  void main()  
03  {  
04      FILE *fp1,*fp2;  
05      char c;  
06      fp1=fopen("a.txt","r");          /*以读的方式打开 a.txt 文件*/  
07      fp2=fopen("b.txt","w");          /*以写的方式打开 b.txt 文件*/  
08      while(!feof(fp1))                /*判断 fp1 是否移动到文件末尾*/  
09      {  
10          c=fgetc(fp1);                 /*通过 fgetc() 函数获取字符至变量 c 中*/  
11          putchar(c);                   /*输出字符变量 c 至屏幕*/  
12      }  
13      rewind(fp1);                      /*调用 rewind() 函数使 fp1 指向文件起始位置*/  
14      while(!feof(fp2))  
15      {  
16          c=fgetc(fp1);  
17          fputc(c,fp2);                  /*将字符 c 写至 fp2 所指文件中*/  
18      }  
19      fclose(fp1);  
20      fclose(fp2);  
21  }
```

- 【代码分析】本例为简单的文件操作范例，详细代码分析如下：
- 第 6 行，以读的方式打开 a.txt 文件，即不能改变文件中的内容。
 - 第 7 行，以写的方式打开 b.txt 文件，因为要将 a.txt 文本中的内容复制到 b.txt 中。
 - 第 8~12 行，利用 feof()函数判断文件指针是否移动到文件末尾，从而输出文件中的所有内容。
 - 第 13 行，调用 rewind()函数，使文件指针 fp1 重新移动到文件的起始位置。
 - 第 14~18 行，通过 while 循环和 feof()函数将 a.txt 中的所有内容复制到 fp2 所指的文件中，即 b.txt 文件。
- 【运行结果】a.txt 文本文件中内容如下：

```
My C File Program!
```

输出结果：My C File Program!

```
b.txt 文本中的内容：My C File Program!
```

12.3.3 fseek()函数和文件随机存取

C 语言中对文件进行读写操作，可以按照顺序方式读写，也可以随机进行读写。关键决定于指向文件的指针位置。若位置指针是按字节有规律的移动，则为顺序读写，若文件位置指针可以移动到文件的任意位置，则称为文件的随机读写。

随机存取是指读写完一个字符后，不一定要读写下一个字符，可以人工地控制文件指针的位置。随机存取可以读写任意位置上的字符，而 C 语言中提供 fseek()函数即可实现随机存取的功能。

调用形式：

```
fseek(文件指针,移动量,起始位置);
```

功能：使位置指针指向固定的位置。

其中文件指针指向要操作的文件。移动量指以起始位置为起点移动的距离，当其值为正值时表示向前移动，其值为负值时表示向后移动。起始位置的值可以为 0、1 和 2，0 表示文件起始位置，1 表示当前文件位置，2 表示文件末尾位置。

起始位置具体情况如表 12-2 所示。

表 12-2 文件起始位置表

起始位置	标识名	数值表示
文件起始	SEEK_SET	0
文件当前位置	SEEK_CUR	1
文件末尾	SEEK_END	2

- 例如：
- fseek(fp,10L,0)：将文件位置指针移动到文件头向前移动 10 个字节处。
 - fseek(fp,20L,1)：将文件位置指针移动到当前位置向前移动 20 个字节处。
 - fseek(fp,-40L,2)：将位置指针移动到文件末尾向后移动 40 个字节处。



【范例 12.9】现磁盘文件 data.txt 中有 10 个学生的信息，利用 fseek()函数将第 1、3、5、7、9 个学生的信息输出至屏幕。

分析：要输出 10 名学生中第 1、3、5、7、9 个学生的信息，可以通过 fseek()函数改变位置指针的指向，再进行输出。

范例 12.9 代码实现

```
01  #include <stdio.h>
02  struct st
03  {
04      char name[20];
05      int age;
06      char sex;
07      float score;
08  }s[10];
09  void main()
10  {
11      FILE *fp;
12      int i;
13      if((fp=fopen("data.txt","w"))==NULL)
14      {
15          printf("打开文件出错\n");
16          exit(1);
17      }
18      for(i=1;i<10;i=i+2)
19      {
20          fseek(fp,i*sizeof(struct st),0);          /*调用 fseek()函数移动到固定位置*/
21          fread(&s[i],sizeof(struct st),1,fp);      /*从文件中读取数据至变量 s[i]中*/
22          printf("%s,%d,%c,%f\n",s[i].name,s[i].age,s[i].sex,s[i].score);
23                                          /*输出结构体中的信息至屏幕*/
24      }
25      fclose(fp);
26  }
```

【代码分析】本例利用 fseek()函数移动指针位置，详细代码分析如下：

- 第 20 行，利用 fseek()函数每次从文件开头移动 i*sizeof(struct st)个字节，其中 i 的值为别 1、3、5、7、9。

【运行结果】data 文本文件中的内容如下：

```
zhangho,20,w,89.5
wangli,19,w,80.5
wangdo,21,m,87.5
lili,22,w,90.0
liuho,21,m,78.5
dinglile,23,m,79.0
xiaofan,22,m,79.5
```



```
caini,21,w,84.0
xiudo,22,m,85.5
xueming,23,m,88.0
```

输出结果如下:

```
wangli,19,w,80.5
lili,22,w,90.0
dinglile,23,m,79.0
caini,21,w,84.0
xueming,23,m,88.0
```

12.3.4 ftell()函数

调用形式:

```
ftell(文件指针);
```

功能: 用于获取文件指针当前的位置, 其值为从文件起始处的位移量。例如:

```
ftell(fp);
```

用于获取文件指针 `fp` 当前的位置。`ftell()`函数也有返回值, 若函数调用出错, 则返回值-1, 若函数正常调用, 则返回文件指针当前的位置。

【范例 12.10】现有一磁盘文件 `file.txt`, 其内容如下:

```
Hello everybody!
Hello everyone!
```

读取文件中的内容并显示在屏幕上, 然后移动文件指针至文件末尾并显示文件指针的位置, 再向文件中添加如下内容:

```
This is a c program!
```

分析: 向文件中添加内容, 应该用追加的方式打开文件, 可移动文件指针的位置到文件的末尾, 再在文件中写入数据。

范例 12.10 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      FILE *fp;
05      long p;
06      if((fp=fopen("file.txt","a+"))==NULL) /*以追加读写的方式打开 file.txt 文本*/
07      {
08          printf("打开文件出错\n");
09          exit(1);
10      }
```



```
11  p=ftell(fp);                /*调用 ftell()函数求文件指针当前位置*/
12  printf("当前指针位置为%d\n",p);    /*输出文件指针当前位置*/
13  fseek(fp,0,2);                /*文件指针移动到文件的末尾*/
14  p=ftell(fp);                /*调用 ftell()函数求文件指针当前位置*/
15  printf("当前指针位置为%d\n",p);
16  fputs("This is a c program!",fp);    /*向 fp 所指文件中追加内容*/
17  p=ftell(fp);
18  printf("当前指针位置为%d\n",p);
19  fclose(fp);
20  }
```

【代码分析】本例向文件中添加内容，详细代码分析如下：

- 第 6 行，用追加的方式打开文件，因为要向文件的末尾添加数据。
- 第 13 行，通过 `fseek()` 函数将文件指针 `fp` 移动到文件的末尾。

【运行结果】`file.txt` 文件中的内容如下：

```
Hello everybody!
Hello everyone! This is a c program!
```

屏幕输出结果如图 12-1 所示。

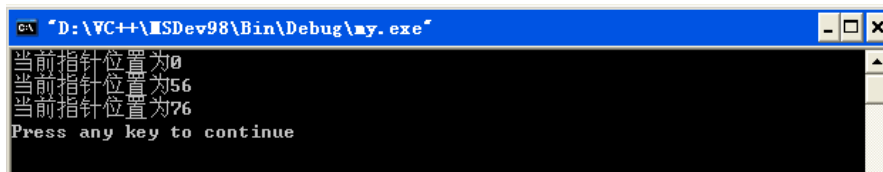


图 12-1 范例 12.10 结果图



12.4 出错检测函数

C 语言不仅提供了大量对文件进行读写操作的函数，还提供了检测是否出错的函数。在本节将重点讲述这些函数。

12.4.1 `ferror()` 函数

调用形式：

```
ferror(文件指针);
```

功能：用于测试调用函数是否出错。例如：

```
ferror(fp);
```

若 `ferror()` 函数的返回值为 0，则表示没出错。若返回非 0 值，则表示出错。每次调用一个函数时，系统都会生成一个新的 `ferror()` 函数值，因此调用函数之后就必须通过 `ferror()` 函数检测返回值。



12.4.2 clearerr()函数

调用形式:

```
clearerr(文件指针);
```

功能: 将文件错误标识和文件结束符置 0。例如:

```
clearerr(fp);
```

若调用一个函数出错, 则其返回值为非 0 值, 通过调用 `clearerr()` 函数即可将该值置为 0。



12.5 程序应用举例

【范例 12.11】 将磁盘中的一个文件内容复制到另一个文件中。

分析: 将一个文件复制到另一个文件中, 则一个文件以读方式打开, 另一个文件应以写方式打开。再通过 `feof()` 函数将文件中所有内容复制到另一个文件中。

范例 12.11 代码实现

```
01  #include <stdio.h>
02  void main()
03  {
04      FILE *fp1,*fp2;
05      char ch,in[10],out[10];
06      printf("输入要复制的文件名:");
07      scanf("%s",in);                      /*获取要进行复制的文件名*/
08      printf("输入复制到哪一个文件:");
09      scanf("%s",out);
10      if((fp1=fopen(in,"r"))==NULL)        /*以读的方式打开变量 in 对应的文件*/
11      {
12          printf("不能打开文件\n");
13          exit(0);
14      }
15      if((fp2=fopen(out,"w"))==NULL)       /*以写的方式打开变量 out 对应的文件*/
16      {
17          printf("不能打开文件\n");
18          exit(0);
19      }
20      while(!feof(fp1))                    /*while 循环一直执行直到文件的末尾*/
21      {
22          ch=fgetc(fp1);                   /*通过 fgetc() 函数从文件中获取字符至变量 ch*/
23          fputc(ch,fp2);
24      }
25      fclose(fp1);
```




```
26     fclose(fp2);
27 }
```

【代码分析】本例复制文件中的内容，详细代码分析如下：

- 第 20~24 行，利用 `feof()` 函数判断是否是文件末尾，通过 `fputc()` 函数将字符逐个写入要写入的文件中。

【运行结果】输入要复制的文件名:file1.txt。

输入要复制到哪一个文件:file2.txt。

file1.txt 文本中的内容如下：

```
Hello everyone!
This is a c program!
there is a duck!
```

file2.txt 文本中的内容如下：

```
Hello everyone!
This is a c program!
there is a duck!
```

【范例 12.12】现有一文件记录了 10 名学生的信息，其中包括姓名、学号及 4 门科目的成绩，要求从文件中读取出数据并计算 4 门科目的平均成绩，写至另外一个文件中。

分析：从文件中读取数据可以利用 `fread()` 或 `fscanf()` 函数来实现，将数据写至文件中可以使用 `fwrite()` 或 `fprintf()` 函数。

范例 12.12 代码实现

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  void main()
04  {
05      void f();                                /*自定义函数 f() 声明*/
06      char s[10]={"学号","姓名","语文","数学","英语","计算机","平均成绩"};
07      f(s);                                    /*调用自定义函数 f()*/
08  }
09  void f(char (*p)[10])
10  {
11      FILE *fp;
12      int score[10][4];
13      float aver[10];
14      char xuehao[10][10],name[10][20];
15      char t[5];
16      int i,j,k=0;
17      float a[10],*p1;
18      p1=a;                                    /*使指针 p1 指向数组 a*/
19      fp=fopen("stuin.txt","r");
20      for(i=0;i<10;i++)
```



```
21  {
22      fscanf(fp, "%s", xuehao[i]);          /*从文件中读取学号至 xuehao[i]中*/
23      fscanf(fp, "%s", name[i]);
24      for(j=0; j<4; j++)
25          fscanf(fp, "%f", score[i][j]);
26      *(p1+i)=(score[i][0]+ score[i][1]+ score[i][2]+ score[i][3])/4.0;
                                   /*计算每名学生的平均分*/
27  }
28  fclose(fp);
29  fp=fopen("stuout.txt", "w");
30  for(i=0; i<10; i++)
31  {
32      fprintf(fp, "%10s", xuehao[i]);        /*将 xuehao[i]写至 fp 所指文件中*/
33      fprintf(fp, "%10s", name[i]);
34      for(j=0; j<4; j++)
35          fprintf(fp, "%3d", score[i][j]);
36      fprintf(fp, ".1f%f", *(p1+i));
37  }
38  fclose(fp);
39  fp=fopen("stuout.txt", "r");
40  for(i=0; i<10; i++)
41  {
42      fscanf(fp, "%s", xuehao[i]);          /*将 stuout.txt 中的学号写入 xuehao[i]中*/
43      fscanf(fp, "%s", name[i]);
44      for(j=0; j<4; j++)
45          fscanf(fp, "%d", score[i][j]);
46      fscanf(fp, "%f", a[i]);
47  }
48  for(i=0; i<10; i++)                  /*输出每名学生的信息*/
49      printf("%10s%10s%3d%3d%3d%3d%6.1f\n", xuehao[i], name[i], score[i][0],
50              score[i][1], score[i][2], score[i][3], a[i]);
51  fclose(fp);
52  }
```

【代码分析】本例读取文件中的内容并进行计算，详细代码分析如下：

- 第 20~27 行，将文件中的数据写至相应的变量中。

【运行结果】stuin.txt 中的内容如下：

```
100781 zhangho  70 89 78 86
100782 wangli   80 87 81 85
100783 wangdo   87 89 90 91
100784 lili     81 80 77 79
100785 liuho    78 79 81 80
100786 dinglile  88 89 93 77
100787 xiaofan  76 75 79 82
100788 caini    84 87 81 79
100789 xiudo    85 87 89 90
```



```
100790 xueming 81 94 79 84
```

stuout.txt 中的内容如下:

```
100781 zhangho 70 89 78 86 80.7
100782 wangli 80 87 81 85 83.2
100783 wangdo 87 89 90 91 89.2
100784 lili 81 80 77 79 79.2
100785 liuho 78 79 81 80 79.5
100786 dinglile 88 89 93 77 86.7
100787 xiaofan 76 75 79 82 78.0
100788 caini 84 87 81 79 82.7
100789 xiudou 85 87 89 90 87.7
100790 xueming 81 94 79 84 84.5
```



12.6 小结

文件是 C 语言的重点概念,是学习 C 语言必须要掌握的知识。本章讲解了文件的基本概念及其操作方法,其中包括文件的读写等。对文件的操作一般都是通过 C 语言提供的库函数来实现的,例如 `fwrite()` 函数可以向文件中写入数据块, `fread()` 函数可以读取文件中的数据块。读者应认真掌握这些库函数的概念及其应用。



12.7 习题

一、选择题

1. 现有以下程序,则下列说法正确的是 ()。

```
FILE *fp;
fseek(fp,0,2);
p=ftell(fp);
printf("%d",p);
```

- A. `fp` 指向文件的开头
- B. `fp` 指向文件的当前位置
- C. `p` 指向当前指针位置,为指针类型
- D. `p` 指向当前指针位置,为整型

【提示】上述程序利用 `ftell()` 函数计算文件指针的当前位置,然后输出。

2. 要打开一个名为“file.txt”的文本文件并在其中写内容,则下列语句正确的是 ()。

- A. `fp=fopen("file.txt","r");`
- B. `fp=fopen("file.txt","a");`
- C. `fp=fopen("file.txt","w");`
- D. `fp=fopen("file.txt","rb");`

【提示】向文件中写内容,则应以写或读写的方式打开。



3. 下列语句是追加方式打开文件的是 ()。

- A. `fp=fopen("a.txt","w+");` B. `fp=fopen("a.txt","w");`
C. `fp=fopen("a.txt","r+");` D. `fp=fopen("a.txt","a");`

【提示】在 C 语言中，以追加的方式打开用“a”表示。

4. 以下程序运行结果为 ()。

```
void main()
{
    int i,n;
    FILE *fp;
    if((fp=fopen("a.txt","w+"))==NULL)
    {
        printf("打开文件出错");
        exit(0);
    }
    for(i=1;i<=10;i++)
        fprintf(fp,"%3d",i);
    for(i=0;i<5;i++)
    {
        fseek(fp,i*3L,1);
        fscanf(fp,"%d",&n);
        printf("%d",n);
    }
    fclose(fp);
}
```

- A. 1 2 3 4 5 B. 6 7 8 9 10
C. 1 3 5 7 9 D. 2 4 6 8 10

【提示】本题通过 `fprintf()` 函数将变量 `i` 写至文件中，然后利用 `fseek()` 函数及 `fscanf()` 函数从文件中获取数据再输出。

5. 以下程序运行结果为 ()。

```
void main()
{
    FILE *fp;
    char s[]="C Program!";
    fp=fopen("C.txt","w");
    fputs(s,fp);
    while(!feof(fp))
    {
        char ch=getchar();
        printf("%c",ch);
    }
    fclose(fp);
}
```

- A. C B. Program



由浅入深学 C 语言——基础、进阶与必做 430 题

C. C Program

D. C Program!

【提示】本题将字符数组 s 中的内容写至 fp 所指文件中，然后从文件中读取至屏幕。

6. 以下程序的功能为 ()。

```
void main()
{
    FILE *fp;
    fp=fopen("b.txt","w+");
    while(!feof(fp))
    {
        if(fgetc=="#")
        {
            fseek(fp,-1L,1);
            fputc('*',fp);
        }
    }
}
```

A. 向文件中写入字符#

B. 向文件中写入字符*

C. 替换文件中的字符#为字符*

D. 以上叙述都不正确

【提示】本题对文本中的字符逐个判断，若为字符#则替换为*。

7. 以下程序的功能为 ()。

```
void main()
{
    FILE *fp1,*fp2;
    if((fp=fopen("a.txt","w+"))==NULL)
    {
        printf("打开文件出错");
        exit(0);
    }
    if((fp2=fopen("b.txt","w+"))==NULL)
    {
        printf("打开文件出错");
        exit(0);
    }
    while(!feof(fp1))
        fputc(fgetc(fp1),fp2);
    fclose(fp1);
    fclose(fp2);
}
```

A. 清空 a.txt 文件

B. 清空 b.txt 文件

C. 将 a.txt 文件中的内容复制到 b.txt 中

D. 将 b.txt 文件中的内容复制到 a.txt 中

【提示】本题中指针 fp1 指向 a.txt，指针 fp2 指向 b.txt，然后将指针 fp1 所指文件中的内容写至 fp2 所指文件中。



8. 以下程序运行结果为 ()。

```
void main()
{
    int i,n;
    FILE *fp;
    if((fp=fopen("a.txt","w+"))==NULL)
    {
        printf("打开文件出错");
        exit(0);
    }
    for(i=1;i<=10;i++)
        fprintf(fp,"%3d",i);
    for(i=0;i<10;i++)
    {
        fseek(fp,i*3L,1);
        fscanf(fp,"%d",&n);
        fseek(fp,i*3L,1);
        fprintf(fp,"%3d ",n+10);
    }
    for(i=1;i<6;i++)
    {
        fseek(fp,i*6L,1);
        fscanf(fp,"%d",&n);
        printf("%3d",n);
    }
    fclose(fp);
}
```

A. 11 12 13 14 15

B. 11 13 15 17 19

C. 1 2 3 4 5

D. 10 12 14 16 18

【提示】本题利用 `fprintf()` 函数写入数据至 `fp` 所指文件中，然后利用 `fscanf()` 函数和 `fseek()` 函数从 `fp` 所指文件中读取相应数据至屏幕。

二、填空题

1. 已知 `data.txt` 文本中的内容为 `acjaidg`，则以下程序输出结果为_____。

```
void main()
{
    FILE *fp;
    fp=fopen("data.txt","r");
    while(!feof(fp))
    {
        char ch=fgetc(fp);
        fputc(ch);
        fseek(fp,1L,1);
    }
}
```



```
fclose(fp);  
}
```

【提示】本题从文件中获取一个字符输出至屏幕，然后通过 `fseek()` 函数从当前位置向前移动 1 个字节，直到文件末尾退出 `while` 循环。

2. 以下程序运行后，`file.txt` 文件中的内容为_____。

```
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
void main()  
{  
    FILE *fp;  
    char s[10]="abcdjeias";  
    fp=fopen("file.txt","w+");  
    if(fp==NULL)  
        printf("打开文件失败\n");  
    int len=strlen(s);  
    for(i=0;i<len/2;i++)  
        fputc(s[i],fp);  
}
```

【提示】本题利用 `fputc()` 函数将字符数组 `s` 中的一半内容写入 `fp` 所指文件中。

3. 若运行以下程序输入 `abcdef*`，则输出结果为_____。

```
#include <stdio.h>  
void main()  
{  
    FILE *fp;  
    char c;  
    if((fp=fopen("f.txt","w"))==NULL)  
    {  
        printf("打开文件出错\n");  
        exit(0);  
    }  
    c=getchar();  
    while(c!='*')  
    {  
        fputc(c,fp);  
        putchar(c);  
        c=getchar();  
    }  
    fclose(fp);  
}
```

【提示】本题从键盘输入一串字符，将其写入 `fp` 所指文件中并输出至屏幕。

4. 运行以下程序输入 `Hello 15`，则其输出结果为_____。



```
#include <stdio.h>
void main()
{
    char c[50];
    int x;
    FILE *fp;
    if((fp=fopen("f.txt","w"))==NULL)
    {
        printf("打开文件出错\n");
        exit(0);
    }
    scanf("%s%d",c,&x);
    fprintf(fp,"%s%d",c,a);
    fclose(fp);
    if((fp=fopen("f.txt","r"))==NULL)
    {
        printf("打开文件出错\n");
        exit(0);
    }
    fscanf(fp,"%s%d",s,&x);
    printf("%s %d",s,a);
    fclose(fp);
}
```

【提示】本题以写的方式打开文件，将从键盘输入的数据写至文件中，关闭文件指针。然后以读的方式打开文件，从文件中读取数据至屏幕。

5. 以下程序运行结果为_____。

```
void main()
{
    FILE *fp;
    int i;
    char s[][10]={"Beijing","shanghai","tianjin"};
    fp=fopen("a.txt","w");
    if(fp==NULL)
        printf("打开文件出错\n");
    for(i=0;i<3;i++)
        fputs(s[i],fp);
    fclose(fp);
    fp=fopen("a.txt","r");
    if(fp==NULL)
        printf("打开文件出错\n");
    for(i=0;i<3;i++)
    {
        fgets(s[i],10,fp);
        printf("%s",s[i]);
    }
}
```




```
fclose(fp);  
}
```

【提示】本题以写的方式打开文件将二维字符数组 s 写至文件中，然后从文件中读取数据至屏幕。

6. 以下程序运行结果为_____。

```
void main()  
{  
    FILE *fp;  
    int i,n;  
    fp=fopen("a.txt","w");  
    for(i=0;i<10;i++)  
        fprintf(fp,"%2d",i);  
    for(i=0;i<10;i++)  
    {  
        fscanf(fp,"%2d",&n);  
        printf("%2d",n);  
    }  
}
```

【提示】本题以写的方式将 0~9 写至 fp 所指文件中，然后将 0~9 输出至屏幕。

7. 已知 data.txt 文本中的内容为 abadjck，则以下程序运行结果为_____。

```
void main()  
{  
    FILE *fp;  
    int sum=0;  
    fp=fopen("data.txt","r");  
    if(fp==NULL)  
        printf("打开文件出错\n");  
    while(!feof(fp))  
    {  
        char ch=fgetc(fp);  
        if(ch!=EOF)  
            sum++;  
    }  
    printf("文本中字符个数为%d\n",sum);  
    fclose(fp);  
}
```

【提示】本题通过 feof()函数逐个读取文件中的字符，统计字符个数后输出至屏幕。

8. 以下程序运行结果为_____。

```
struct stu  
{  
    char n[10];  
    int score;
```



```
}
void main()
{
    struct stu s[4]={{"Lili",89},{"Wangming",90},{"Yanghua",78},{"Zhangxiao",88}};
    FILE *fp;
    int i;
    fp=fopen("stu.txt","w");
    if(fp==NULL)
        printf("打开文件出错\n");
    for(i=0;i<4;i++)
        fwrite(s+i,sizeof(struct stu),1,fp);
    fclose(fp);
    fp=fopen("stu.txt","r");
    if(fp==NULL)
        printf("打开文件出错\n");
    for(i=0;i<4;i++)
    {
        fread(s+i,sizeof(struct stu),1,fp);
        printf("%s,%d\n",s[i].n,s[i].score);
    }
    fclose(fp);
}
```

【提示】本题先利用写的方式将结构体数组 `s` 写至相应文件中，然后利用读的方式从文件中读取数据至屏幕。

9. 以下程序运行结果为_____。

```
void main()
{
    char *s="abcsade";
    FILE *fp;
    fp=fopen("a.txt","w");
    fputs(s,fp);
    while(!feof(fp))
    {
        char ch=fgetc(fp);
        printf("%c",ch);
    }
    fclose(fp);
}
```

【提示】本题将字符串“abcsade”写至文件中，然后输出至屏幕。

10. 已知 `a.txt` 文件中内容为“aajkcna1”，则程序运行之后 `b.txt` 文件中的内容为_____。

```
void main()
{
    FILE *fp1,*fp2;
```



```
fp1=fopen("a.txt","r");
fp2=fopen("b.txt","w");
while(!feof(fp1))
{
    char ch=fgetc(fp1);
    fputc(ch,fp2);
}
fclose(fp1);
fclose(fp2);
}
```

【提示】上述程序的功能为将 fp1 所指文件中内容写至 fp2 所指文件中。

三、编程题

1. 编写程序，将磁盘文件中的内容显示在屏幕上。

【提示】显示文件中的内容，可以定义一个文件指针指向该文件。再通过对文件指针的操作，逐个输出文件中的内容。

【核心代码】

```
while(!feof(fp))
{
    ch=fgetc(fp);
    putchar(ch);
}
```

2. 编写一个程序，由键盘输入 5 个学生的信息，并将数据保存在名为 student.txt 的文本文件中。

【提示】学生信息包括多种类型数据，因此可以定义为结构体类型。又因为有多个学生，所以定义结构体数组来保存学生信息。从键盘输入学生的信息，保存至相应的结构体变量中，再通过 fwrite() 函数将数据写入文件。

【核心代码】

```
struct st
{
    char name[20];
    int age;
    char sex;
    float score;
}s[5];
if((fp=fopen("student.txt","w"))==NULL)
{
    printf("打开文件出错\n");
    exit(1);
}
for(i=0;i<5;i++)
    scanf("%s%d%c%f",s[i].name,&s[i].age,&s[i].sex,&s[i].score);
```



```
for(i=0;i<10;i++)  
fwrite(&s[i],sizeof(struct st),1,fp);
```

3. 编写一个程序，实现文件的复制功能。

【提示】要将整个文件复制至另一个文件中，可以通过 `feof()` 函数判断是否到文件末尾，逐个字符复制到另一个文件中。

【核心代码】

```
fp1=fopen(in,"r");  
fp2=fopen(out,"w")  
while(!feof(fp1))  
{  
    ch=fgetc(fp1);  
    fputc(ch,fp2);  
}
```

4. 编写程序，实现两个文件的合并，即将两个文件中的内容复制到其中一个文件中。

【提示】要将两个文件的内容合并成一个文件，则应以追加方式打开文件。再利用读写函数将另一个文件中的内容复制到该文件中即可实现文件的合并。

【核心代码】

```
fp1=fopen("file1.txt","a");  
fp2=fopen("file.txt","r");  
while(!feof(fp2))  
{  
    ch=fgetc(fp2);  
    fputc(ch,fp1);  
}
```

5. 编写程序，从文件中隔一个字符进行读取，例如文件中内容为“jajasiddcb”，则输出 jjscd 至屏幕上。

【提示】将文件中的内容输出至屏幕，则应以读的方式打开文件。通过文件指针进行相应的操作再输出字符至屏幕。

【核心代码】

```
fp=fopen("file.txt","r");  
while(!feof(fp))  
{  
    fgetc(ch,fp);  
    printf("%c",ch);  
    fseek(fp,1L,1);  
}
```

第 13 章 图形处理基础知识

数据以图形方式显示，可使数据直观并且容易理解。随着社会的发展，图形程序如今在计算机各个方面应用非常广泛。C 语言提供了强大的绘图功能，包含丰富的有关图形的函数，利用这些函数可以方便快捷地进行图形程序的设计。

本章主要涉及的知识点有：

- 分辨率；
- 图形函数；
- 图形概念；
- 图形的应用。



13.1 C 语言图形基本概念

C 语言中有着大量的图形函数，用来对图形进行操作，所有函数都包含在 `graphics.h` 头文件中。因此在调用之前，必须在程序的开头用 `#include` 命令将 `graphics.h` 头文件包含进来。

在图形模式下，图形是由一个个点组成的，其中屏幕上的每一个点称为像素。屏幕上包含像素的个数称为分辨率，分辨率越高则点个数越多，显示得越细腻，图形越清晰。例如一个屏幕分辨率为 800×600 ，则其水平方向有 800 个像素，垂直方向有 600 个像素。

在图形中，每一个像素点都由一个确定的坐标来表示。在整个坐标系中，屏幕最左上方的点为坐标原点，坐标为 $(0,0)$ ，水平方向为 x 轴，垂直方向为 y 轴。根据分辨率的不同，则坐标系上的像素点个数也不同。

在图形界面中，坐标有两种形式，即绝对坐标和相对坐标。绝对坐标的参考点为坐标 $(0,0)$ ，每一个像素点有不同的绝对坐标。相对坐标的参考点为当前坐标，即其相比较的点不是坐标原点 $(0,0)$ ，而是当前的点。一般坐标形式都采用绝对坐标来表示。

在绘图之前，必须为屏幕图形适配器设置一种图形模式，即图形的初始化。初始化之后，便可对图形进行操作。



13.2 基本图形函数

图形开发虽然不是 C 语言开发的重点，但读者也要了解 C 语言基本的图形函数，并通过本节学习这些基本函数的应用。

13.2.1 图形初始化

不同的显示器图形适配器有着不同的分辨率，即使是同一个显示适配器在不同模式下也有着不同的分辨率。因此在绘图之前，必须设置某一种图形模式。在未设置模式前，系统默认模式为文本模式，不能进行绘图操作。设置图形模式可以通过 `initgraph()` 函数实现，其形式如下：

```
initgraph(类型 *gdriver, 类型 *gmode, 类型 *path);
```

函数 `initgraph()` 的功能为初始化图形，含有三个参数。其中 `gdriver` 表示要装入的图形驱动程序，`gmode` 表示图形的模式，`path` 表示图形驱动所在的路径。

C 语言中包含的图形驱动程序如表 13-1 所示。

表 13-1 图形驱动程序符号常数及其对应数值

符号常数	对应数值
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMONO	5
IBM8514	6
HERCMONO	7
ATT400	8
VGA	9
PC3270	10
DETECT	0

图形模式如表 13-2 所示。

表 13-2 图形模式符号常量及其对应数值

符号常量	对应数值	对应色调
CGAC0	0	C0
CGAC1	1	C1
CGAC2	2	C2
CGAC3	3	C3
CGAHI	4	2 色
MCGAC0	0	C0
MCGAC1	1	C1
MCGAC2	2	C2
MCGAC3	3	C3
MCGAMED	4	2 色
MCGAHI	5	2 色
EGALO	0	16 色
EGAHI	1	16 色



续表

符号常量	对应数值	对应色调
EGA64L0	0	16 色
EGA64HI	1	4 色
EGAMONHI	0	2 色
IBM8514L0	0	256 色
IBM8514HI	1	256 色
HERCMONHI	0	2 色
ATT400C0	0	C0
ATT400C1	1	C1
ATT400C2	2	C2
ATT400C3	3	C3
ATT400MED	4	2 色
ATT400HI	5	2 色
VGA0	0	16 色
VGA1	1	16 色
VGA2	2	16 色

例如要初始化图形为 MCGA 模式，可用如下代码实现：

```
#include <stdio.h>
void main()
{
    int gdriver,gmode;
    gdriver=MCGA;                /*将 MCGA 赋值给变量 gdriver*/
    gmode=ATT400C0;
    initgraph(&gdriver,&gmode,"c:\\tc");    /*初始化图形模式*/
    closegraph();
}
```

13.2.2 关闭图形函数

程序运行结束后，若要结束绘图方式返回至文本方式方便工作，这时可调用 `closegraph()` 函数来关闭绘图方式，其调用形式如下：

```
closegraph();
```

其功能为释放图形存储区内容，退出绘图模式，回到调用 `initgraph()` 之前的模式，即文本模式。调用 `closegraph()` 函数不需要参数。

13.2.3 设置外观函数

在图形中，可以通过 `setviewport()` 函数定义一个视口在屏幕上。视口相当于一个窗口，图形都显示在该窗口内。图形中的坐标都是相对于该窗口而言的，其左上角坐标为 (0,0)。函数 `setviewport()` 分为可裁剪和不可裁剪两种，其调用形式如下：



```
setviewport(x1,y1,x2,y2,c);
```

其中 $x1$, $y1$ 分别为左上角的坐标, $x2$, $y2$ 为右下角的坐标, c 为裁剪参数。若 c 值为 1 表示超出窗口范围则自动裁剪, 若 c 值为 0 表示超出窗口范围不进行裁剪。



注意: 图形函数参数在未进行说明时, 其默认类型为整型。

13.2.4 清除窗口函数

清除窗口是指将当前位置坐标置 (0,0), 即左上角坐标, 因此原窗口将不复存在, 即清除了窗口。可以通过 `clearviewport()` 函数实现, 其调用形式如下:

```
clearviewport();
```

调用该函数时不需要相应的参数。

13.2.5 清屏函数

清理屏幕可以通过函数 `cleardevice()` 实现。该函数将位置坐标变为 (0,0), 不会改变其他的设置, 可以清除整个屏幕中的内容。清屏函数调用形式如下:

```
cleardevice();
```

该函数调用时也不要参数。

13.2.6 绘图函数

C 语言中提供了多种绘图函数, 其中包括绘制直线、矩形、圆形、弧形等, 在下面内容中将会一一讲解。

1. 绘制线函数

在屏幕上画一条直线可以通过函数 `line()` 实现, 其调用形式如下:

```
line(x1,y1,x2,y2);
```

其中 $x1$, $y1$ 表示起点的坐标, $x2$, $y2$ 表示终点的坐标。

2. 绘制矩形函数

在屏幕上绘制一个矩形可以通过 `rectangle()` 函数实现, 其调用形式如下:

```
rectangle(x1,y1,x2,y2);
```

其中 $x1$, $y1$ 为矩形左上角点的坐标, $x2$, $y2$ 为矩形右下角点坐标。

3. 绘制字符串函数

库函数中的 `outtextxy()` 可以在屏幕某一位置输出特定的字符串, 其调用形式如下:

```
outtextxy(x,y,"特定字符串");
```




其中 x , y 表示字符串对应的坐标位置, 字符串为一串字符, 表示要输出至屏幕的字符串。

4. 绘制矩形块函数

`bar()`函数可以用来绘制一个颜色块。块内的颜色可以用指定的颜色进行填充, 该函数既可以绘制矩形块, 也可以绘制粗线。其调用形式如下:

```
bar(x1,y1,x2,y2);
```

其中 $x1$, $y1$ 分别为矩形块左上角坐标, $x2$, $y2$ 分别为矩形块右下角坐标。

5. 绘制圆函数

`circle()`函数可以在屏幕上绘制一个圆, 其调用形式如下:

```
circle(x,y,r);
```

其中 x 和 y 表示圆心的坐标, r 表示圆的半径。

6. 绘制弧函数

`arc()`函数可以在屏幕上绘制一条弧, 其调用形式如下:

```
arc(x,y,a1,a2,r);
```

其中 x 和 y 表示弧圆心的坐标, $a1$ 和 $a2$ 分别表示弧的起始角和终止角, r 表示弧的半径。
`arc()`函数是以 x 和 y 为圆心, r 为半径, $a1$ 为起始角度, $a2$ 为终止角度来绘制一条弧的。

7. 绘制画椭圆函数

`ellipse()`函数可以在屏幕上绘制一个椭圆, 其调用形式如下:

```
ellipse(x,y,a1,a2,a,b);
```

其中 x 和 y 表示椭圆的中心坐标, $a1$ 为起始角度, $a2$ 为终止角度, a 表示长半轴的长度, b 表示短半轴的长度。

8. 绘制扇形函数

在屏幕上绘制一个扇形可以通过 `pieslice()`函数实现, 其调用形式如下:

```
pieslice(x,y,a1,a2,R);
```

其中 x 和 y 表示扇形圆心坐标, $a1$ 为起始角度, $a2$ 为终止角度, R 为扇形的半径长度。

9. 绘制三维矩形函数

`bar3d()`函数可以在屏幕上绘制一个三维的矩形图, 并且可以用特定的颜色和模式进行填充, 其调用形式如下:

```
bar3d(x1,y1,x2,y2,m,n);
```

其中 $x1$ 和 $y1$ 分别为三维矩形左上角点坐标, $x2$ 和 $y2$ 分别为三维矩形右下角坐标, m 表示图形的深度, n 值确定是否需要三维顶。若 n 值为 1 表示有三维顶, 若 n 值为 0 表示无三维顶。



10. 绘制椭圆扇形函数

`sector()`函数可以用来绘制一个椭圆扇形，并用特定颜色填充图形。其调用形式如下：

```
sector(x,y,a1,a2,a,b);
```

其中 x 和 y 表示椭圆的中心坐标， $a1$ 表示椭圆扇形的起始角度， $a2$ 表示椭圆扇形的终止角度， a 表示长半轴的长度， b 表示短半轴的长度。

11. 绘制多边形函数

`drawpoly()`函数可以在屏幕上绘制一个多边形，其调用形式如下：

```
drawpoly(k,s);
```

其中 k 表示多边形顶点的个数， s 表示多边形顶点的集合，即各个顶点的坐标值集合。通常用数组来保存点的集合。

12. 绘制线到指定点函数

`lineto()`函数可以在屏幕上从某一点直线到另外一点，并且改变当前位置坐标。调用该函数后，当前点坐标移动到直线的终点，其调用形式如下：

```
lineto(x,y);
```

其中 x 为起始点， y 为终止点，都为整型。

13. 移动当前点函数

函数 `moveto()`可以将当前点的坐标移动到某一特定位置，其调用形式如下：

```
moveto(x,y);
```

其中 x 和 y 表示要移动到的点的坐标，都为整型。

14. 填充函数

`floodfill()`函数可以用某种颜色填充一片区域，其调用形式如下：

```
floodfill(x,y,n);
```

其中 x 和 y 表示填充区域的点坐标。区域中任何一点都可以，若 x 和 y 在区域内部，则填充整个区域内部，若 x 和 y 在区域外部则填充整个区域的边界处， n 表示要填充的颜色。

15. 椭圆填充函数

`fillellipse()`函数可以用某种颜色来填充椭圆，其调用形式如下：

```
fillellipse(x,y,a,b);
```

其中 x 和 y 为椭圆中心坐标， a 和 b 分别为长半轴和短半轴的长度。

16. 线型函数

在屏幕中绘制的线的粗细、样式可以用 `setlinestyle()`函数来改变。其调用形式如下：



```
setlinestyle(m,n,s);
```

其中 m 为绘制的线参数， n 为定义线型参数， s 为线的粗细参数。
绘制表的线参数如表 13-3 所示。

表 13-3 线参数表

标识符	对应数值	描 述
SOLID_LINE	0	用实线绘制
DOTTED_LINE	1	用点线绘制
CENTER_LINE	2	用中心线绘制
DASHED_LINE	3	用虚线绘制
USERBIT_LINE	4	用户自定义线型

线粗细参数如表 13-4 所示。

表 13-4 线粗细参数表

标识符	对应数值	描 述
NORM_WIDTH	1	宽度为 1 个像素
THICK_WIDTH	3	宽度为 3 个像素

例如绘制一条虚线，可以用如下代码实现：

```
setlinestyle(4,0xF0F0,1);
```

17. 设置背景色函数

setbkcolor()函数可以用来设置屏幕上绘图背景颜色，其调用形式如下：

```
setbkcolor(c);
```

其中 c 为整型，表示背景颜色的值，既可以为整型常数，也可以字符型常数。
背景颜色参数如表 13-5 所示。

表 13-5 背景颜色参数表

字符常数	数 值	对应颜色
BLACK	0	黑
BLUE	1	蓝
GREEN	2	绿
CYAN	3	青
RED	4	红
MAGENTA	5	紫红
BROWN	6	棕
LIGHTGRAY	7	浅灰
DARKGRAY	8	深灰
LIGHTBLUE	9	浅蓝
LIGHTGREEN	10	浅绿
LIGHTCYAN	11	浅青

续表

字符常数	数 值	对应颜色
LIGHTRED	12	淡红
LIGHTMAGENTA	13	淡紫
YELLOW	14	黄
WHITE	15	白

18. 前景色函数

setcolor()函数可以用来改变前景色，即当前绘画的颜色，其调用形式如下：

```
setcolor(c);
```

其中 c 表示前景的颜色。

19. 设置样式及颜色函数

setfillstyle()函数可以用来设置填充模式及其颜色，其调用形式如下：

```
setfillstyle(pat,c);
```

其中 pat 表示填充采取的模式，c 表示填充采取的颜色。填充模式包括 12 种，如表 13-6 所示。

表 13-6 填充的模式表

字符常数	对应数值	意 义
EMPTY_FILL	0	背景色填充
SOLID_FILL	1	特点颜色填充
LINE_FILL	2	线性填充
LISLASH_FILL	3	斜线填充
SLASH_FILL	4	粗斜线填充
BKSLASH_FILL	5	反的粗斜线填充
LTBKSLASH_FILL	6	反的斜线填充
HATCH_FILL	7	网线填充
XHATCH_FILL	8	斜交叉的线填充
INTERLEAVE_FILL	9	间隔的线填充
WINDEDOT_FILL	10	稀疏的点填充
CLOSEDOT_FILL	11	密集的点填充
USER_FILL	12	自定义模式填充



13.3 图形应用范例

【范例 13.1】通过下面的例子，简单了解绘图函数的使用。

分析：C 语言中包含大量的绘图函数，如绘制椭圆，矩形等图形，同时也有设置屏幕颜色等的函数，分别具有不同的功能。



范例 13.1 代码实现

```
01  #include <stdio.h>
02  #include <graphics.h>          /*包含 graphics.h 头文件*/
03  void main()
04  {
05      int gd,gm,i;
06      gd=DETECT;                  /*将 DETECT 赋值给变量 gd*/
07      initgraph(&gd,&gm,"c:\\tc"); /*调用 initgraph()函数初始化图形*/
08      setbkcolor(2);              /*设置背景颜色*/
09      cleardevice();              /*清屏*/
10      for(i=0;i<16;i++)
11      {
12          setcolor(i);            /*设置颜色*/
13          circle(300,250,30+10*i); /*调用 circle()函数绘圆*/
14          delay(200);             /*延时 200 毫秒*/
15      }
16      closegraph();              /*关闭图形绘画模式*/
17  }
```

【代码分析】本例为图形简单范例，详细代码分析如下：

- 第 7 行，利用 `initgraph()` 函数初始化图形。
- 第 10~15 行，分别用不同的颜色来绘制圆形。
- 第 16 行，调用 `closegraph()` 函数关闭图形绘画模式。

【运行结果】上述程序将会以不同的颜色绘制同一个圆，该圆的中心坐标为（300,250）。

【范例 13.2】绘制一个矩形，并指定样式填充。

分析：绘制矩形可以通过 `rectangle()` 函数实现，填充则可通过 `setfillstyle()` 函数实现。

范例 13.2 代码实现

```
01  #include <stdio.h>
02  #include <graphics.h>
03  void main()
04  {
05      int gm=0,gd=DETECT;          /*初始化图形模式以及驱动*/
06      initgraph(&gd,&gm,"d:\\tc"); /*调用 initgraph()函数初始化图形*/
07      rectangle(100,200,400,500); /*绘制矩形*/
08      setfillstyle(1,2);           /*设置模式*/
09      floodfill(100,200,2);        /*填充样式*/
10      closegraph();               /*关闭图形绘画模式*/
11  }
```

【代码分析】本例在屏幕上绘制矩形，详细代码分析如下：

- 第 7 行，利用 `rectangle()` 函数在屏幕上绘制一个矩形。
- 第 8、9 行，用特定样式填充绘制的矩形。

【运行结果】程序运行后在屏幕上出现一个矩形，起点为（100,200），终点为（400,500）。



13.4 小结

在本章中讲解了图形的概念及其操作函数，图形技术在计算机中应用的非常广泛。图形操作函数一般被包含在 `graphics.h` 头文件中，因此在调用之前必须先包含该头文件。初学者只需了解本章讲解的图形基本操作函数概念及其应用即可。



13.5 习题

一、填空题

1. 下列程序的功能为在屏幕上绘制一个三角形，补全下面的代码。

```
#include <graphics.h>
void main()
{
    int gd=DETECT,gm=1;
    int x1,y1,x2,y2,x3,y3;
    initgraph(_____,_____, "d:\\tc");
    cleardevice();
    x1=50;
    y1=50;
    x2=100;
    y2=100;
    x3=70;
    y3=90;
    lineto(x1,y1,x2,y2);
    lineto(_____);
    lineto(x2,y2,x3,y3);
    closegraph();
}
```

【提示】`initgraph()`函数初始化图形时，必须有图形的驱动 `gdriver` 及图形模式 `gmode`。

二、编程题

1. 从键盘输入一个半径 r ，以 r 为半径绘制一个圆形。

【提示】在屏幕上绘制一个图形，必须先通过 `initgraph()`函数初始化，再通过相应的函数绘制相应的图形。

【核心代码】

```
#include <graphics.h>
void main()
{
```



```
initgraph(&gdriver,&gmode,"c:\\\\tc");
scanf("%d",&r);
cleardevice();
circle(x,y,r);
}
```

2. 编写一个程序，在屏幕上绘制一个矩形，并用不同的样式进行填充。

【提示】绘制矩形可以通过 `rectangle()` 函数实现，用特定的样式填充则可以调用 `setfillstyle()` 函数实现。

【核心代码】

```
initgraph(&gdriver,&gmode,"c:\\\\tc");
cleardevice();
for(i=0;i<10;i++)
{
    rectangle(x,y,x+100,y+150);
    setfillstyle(i,i);
    floodfill(x,y);
}
closegraph();
```

3. 输入两个点的坐标，以这两个点为起始点和终止点在屏幕上绘制一条直线。

【提示】将从键盘输入的数保存至相应的变量中，然后调用 `line()` 函数可以在两点之间绘制一条直线。

【核心代码】

```
initgraph(&gdriver,&gmode,"c:\\\\tc");
cleardevice();
scanf("%d%d%d%d",x1,y1,x2,y2);
line(x1,y1,x2,y2);
```

第 14 章 预处理宏命令

预处理是指程序在被编译前，预处理器对程序代码进行处理的过程。C 语言程序在编译之前，都会首先经过预处理，大多数 C 编译软件都包含预处理程序。

在 C 语言中预处理程序中主要分析和处理以“#”开头的代码行。预处理包含宏、文件包含、条件编译等，通过预处理可以提高程序的可读性、条理性，而且易于修改。

程序经过预处理过程后，预处理命令将不再存在，例如宏命令会被相应的表达式替换。最后整个程序经编译程序编译后，生成可执行目标代码文件。

本章主要涉及的知识点有：

- 宏定义；
- 宏的应用；
- 文件的包含；
- 条件编译；
- 变量类型。



14.1 宏

宏定义指用一个用户标识符表示一个字符串，对文件进行预处理的过程中，宏名将会被宏定义的字符串所替换。宏分为不带参数的宏和带参数的宏，本节中将重点讲解这两种宏的概念及其应用。

14.1.1 不带参数的宏

宏定义形式如下：

```
#define 用户标识符 字符串
```

宏定义包含以下三种方式。

(1) #define 用户标识符 常量。例如：

```
#define x 5
```

定义一个常量，即 x 的值为 5。

(2) #define 用户标识符 表达式。例如：

```
#define x 3*y-1
```

表示 x 出现的地方用表达式 3*y-1 替换。

(3) #define 用户标识符 字符串常量。例如：



```
#define x string
```

表示变量 `x` 出现的地方用字符串 `string` 替换。



注意：

- (1) 为了与变量名区别开，一般用大写字母表示宏名
- (2) 宏名一般取较短或有特殊意义的名字，便于理解和使用。

【范例 14.1】通过下面的例子，简单了解宏定义。

分析：宏定义可以表示常量、表达式、字符串等，可以使程序简洁条理化，修改起来也很方便，只需要在宏定义处修改数值即可。

范例 14.1 代码实现

```
01  #include <stdio.h>
02  #define PI 3.1415926          /*宏定义PI 表示数值 3.1415926*/
03  void main()
04  {
05      float r,area;
06      printf("输入圆的半径:");
07      scanf("%f",&r);
08      area=PI*r*r;
09      printf("圆的面积为%.3f\n",area);
10  }
```

【代码分析】本例为宏定义的简单范例，详细代码分析如下：

- 第 2 行，定义了一个宏。其中宏名为 `PI`，其值为 `3.1415926`。
- 第 8 行，使用宏名参与计算，其中 `PI` 将会被值 `3.1415926` 替换。

【运行结果】该程序的执行结果如图 14-1 所示。

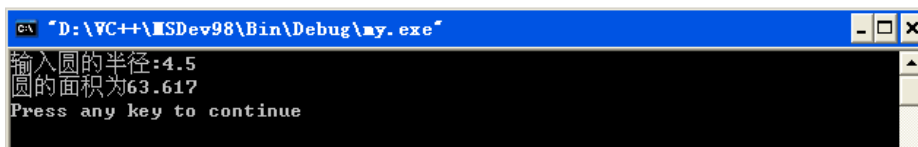


图 14-1 范例 14.1 结果图

【范例 14.2】利用宏命令，计算圆的周长、面积和体积。

分析：计算圆的周长可以利用公式 $L=2 \times 3.14 \times r$ 求解，面积通过公式 $S=3.14 \times r \times r$ 计算，体积通过公式 $V=4 \times 3.14 \times r^3 / 3$ 计算。

范例 14.2 代码实现

```
01  #include <stdio.h>
02  #define P printf              /*宏定义P 表示字符串 printf*/
03  #define PI 3.1415926         /*宏定义PI 表示值 3.1415926*/
04  void main()
```



```
05  {
06    float r,s,l,v;
07    P("输入圆的半径:");
08    scanf("%f",&r);
09    l=2*PI*r;                      /*计算圆的周长*/
10    s=PI*r*r;
11    v=4*PI*r*r*r/3;
12    P("圆的周长为:%.3f\n",l);      /*这里利用不带参的宏 P，输出了圆周长*/
13    P("圆的面积为:%.3f\n",s);
14    P("圆的体积为:%.3f\n",v);
15 }
```

【代码分析】本例利用宏替换字符串，详细代码分析如下：

- 第 7 行，宏名 P 将会被字符串 printf 取代，即调用 printf() 函数。

【运行结果】该程序的执行效果见图 14-2 所示。

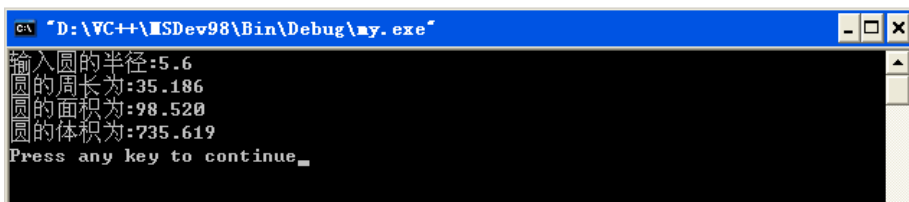


图 14-2 范例 14.2 结果图



注意：

(1) 程序预处理过程只会将宏替换，并不会做相应的检查。例如在定义时#define PI 3.1415926 中将 1 写成字母“l”，预处理程序依旧会进行替换，只有编译出错才会提示。

(2) 宏定义不是语句，不需要在后面加分号。如果有分号，表示宏名替换时包括后面的分号。

例如：#define x 5;

y=x*x;经替换为 y=5;*5;;

显然语法错误，编译时会出错。

(3) 宏定义可以进行层叠定义。

(4) 宏定义 define 作用域为从定义位置到文件结尾。若要限制宏的作用域，可以用#undef 关键字取消宏定义。

例如：

```
#define x 8
...
#undef x
...
```

经过#undef 命令之后，宏名 x 不再表示数值 8。

(5) 宏定义可以表示字符串。



【范例 14.3】改写范例 14.2，通过层叠宏定义求圆的面积和体积。

分析：层叠宏定义指后面的宏定义中包含了前面的宏名，宏名中嵌套着另外一个宏名。

范例 14.3 代码实现

```
01  #include <stdio.h>
02  #define PI 3.1415926
03  #define S PI*r*r          /*宏定义 S 表示 PI*r*r*/
04  #define V 4*PI*r*r*r/3    /*宏定义 V 表示 4*PI*r*r*r/3 */
05  void main()
06  {
07      float r;
08      scanf("%f",&r);        /*从键盘获取数据至变量 r 中*/
09      printf("%.3f,%.3f\n",S,V); /*利用宏 S、V 输出圆的面积和体积*/
10  }
```

【代码分析】本例为宏应用范例，详细代码分析如下：

- 第 2~4 行为层叠宏定义，宏定义 S 和 V 中都包含了宏 PI。

【运行结果】该程序的执行结果如图 14-3 所示。

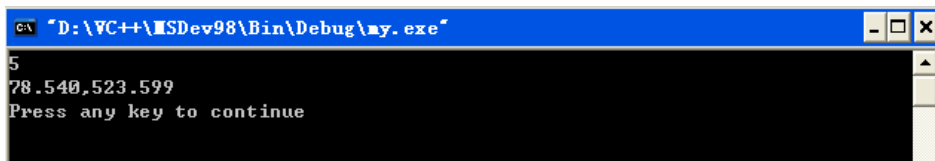


图 14-3 范例 14.3 结果图

14.1.2 带参数的宏

带参数的宏即宏中含有参数，其定义形式如下：

```
#define 宏名 字符串
```

其中字符串应含有对应的参数。例如：

```
#define S(x,y) x*y
#define H(x,y) sqrt(x*y)
s=S(4,5);
printf("%f",H(5,6));
```

以上程序分别为已知长方形的长 x 和宽 y ，求长方形的面积 s ，以及已知直角三角形的两边 x 和 y ，求其斜边的长。

带参数的宏使用方式与带参数的函数类似，调用时要有对应的参数，否则会出错。

【范例 14.4】通过下面的例子，了解带参数的宏概念。

分析：带参数的宏字符串中参数必须包含宏名中的参数，其调用方式与调用函数方式类似。



范例 14.4 代码实现

```

01  #include <stdio.h>                                /*包含 stdio.h 头文件*/
02  #define L(a,b)  a*a+b*b                            /*宏定义 L(a,b) 表示 a*a+b*b*/
03  #define H(a,b)  a*b                                /*宏定义 H(a,b) 表示 a*b*/
04  void main()
05  {
06      float x,y,z,s;
07      printf("输入两个数:");
08      scanf("%f%f",&x,&y);                            /*从键盘获取数据至变量 x 和 y 中*/
09      z=L(x,y);                                        /*调用宏 L(x,y) 计算*/
10      s=H(x,y)+x;                                    /*调用宏 H(x,y) 运算*/
11      printf("z=%.1f\n",z);
12      printf("s=%.1f\n",s);
13  }

```

【代码分析】本例为带参数的宏应用范例，详细代码分析如下：

- 第 2、3 行，定义两个带参数的宏。
- 第 9、10 行，调用宏 L 和 H，计算得出结果。

【运行结果】该程序的执行结果如图 14-4 所示。

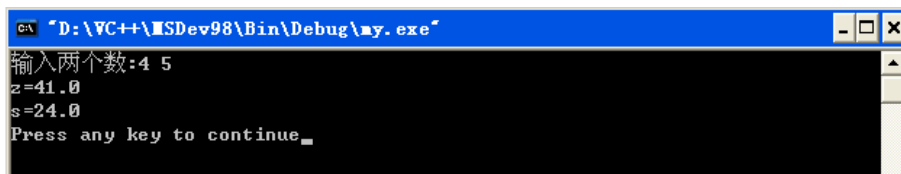


图 14-4 范例 14.4 结果图



注意：

(1) 带参数的宏预编译后只是用#define 命令中的形参来替换的。例如：

```

#define L(x,y)  sqrt(x*x+y*y)
z=L(x,y)改为 z=L(x+1,y+1)
这时替换后 z=sqrt(x+1*x+1+y+1*y+1);

```

显然与实际期望结果不同,要想得到正确结果则 x 和 y 应分别加上括号。如下所示。

```

#define L(x,y)  sqrt((x)*(x)+(y)*(y))

```

(2) 带参数的宏定义名与参数之间不能有空格。

例如：

```

#define L (x,y)  sqrt(x*x+y*y)

```

编译时不会出错，但运行结果会出错。

带参数的宏与带参数的函数使用方式类似，但应注意以下几方面的区别：

- 函数调用参数传递的过程中会产生一个中间值，而带参数的宏调用只是进行字符串的替换，不会产生中间值。



由浅入深学 C 语言——基础、进阶与必做 430 题

- 函数中实参要与形参的类型一致，而宏定义不要求一致。
- 函数只能返回一个值，而宏定义可以返回多个值。

【范例 14.5】通过以下程序，进一步了解宏定义。

分析：带参宏定义可以将文件中多次出现的表达式用宏名替代，使程序更加简洁清晰。

范例 14.5 代码实现

```
01  #include <stdio.h>
02  #define PI 3.1415926
03  #define H(x,y,l,s) l=2*(x+y);s=x*y;          /*宏定义*/
04  void main()
05  {
06      float x,y,l,s;
07      printf("输入两边:");
08      scanf("%f%f",&x,&y);                      /*调用 scanf()函数实现数据的输入*/
09      H(x,y,l,s)                                /*调用宏定义H(x,y,l,s)*/
10      printf("l=%.1f,s=%.1f\n",l,s);
11  }
```

【代码分析】本例为带参数的宏范例，详细代码分析如下：

- 第 3 行为带参数宏定义，即用 $l=2*(x+y);s=x*y$ 替换 $H(x,y,l,s)$ 。

【运行结果】该程序的执行结果如图 14-5 所示。

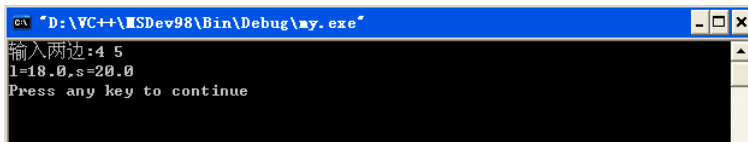


图 14-5 范例 14.5 结果图



14.2 文件包含

文件包含是指在程序中使用 `#include` 命令将一个文件中的内容全部包含进来，而不用将文件中的内容写至程序中。文件包含的形式如下：

```
#include <文件名称> 或 #include "文件名称"
```

例如：

```
#include <stdio.h>
```

表示将 `stdio.h` 头文件中的内容复制到当前的源文件中，因此可以直接调用 `stdio.h` 头文件中已定义好的函数。在 C 语言中以 `.h` 为后缀名的文件都为头文件，其中用引号和尖括号有区别。用引号包围文件系统将只在当前目录下寻找该文件，若寻找失败则再到 C 库函数的头文件目录



下寻找。用尖括号包围文件，编译系统将只到 C 库函数头文件目录下寻找该文件，若未找到该文件，编译系统会报错。

文件包含主要用于将单独的文件集合在同一个文件中，它可以使程序结构化，方便程序员的编程，减少程序所需的内存空间。

【范例 14.6】 通过下面的例子，了解文件包含的概念。

分析：文件包含是指在 C 语言源程序中用 `include` 命令包含相应的文件或用户自定义文件，方便直接调用 C 语言提供的头文件中的库函数或文件中的内容。

范例 14.6 代码实现

(1) f.c 文件中的代码

```
01  #define N  "\n"                                /*宏定义*/
02  #define M  "%4d"
03  #define D  M N
04  #define D1 M M N
05  #define S  "%s" N
06  #define P  printf
```

(2) c.h 头文件

```
01  #define PI 3.1415926
02  #define H int
03  #define K char
```

(3) cfile.c 源代码

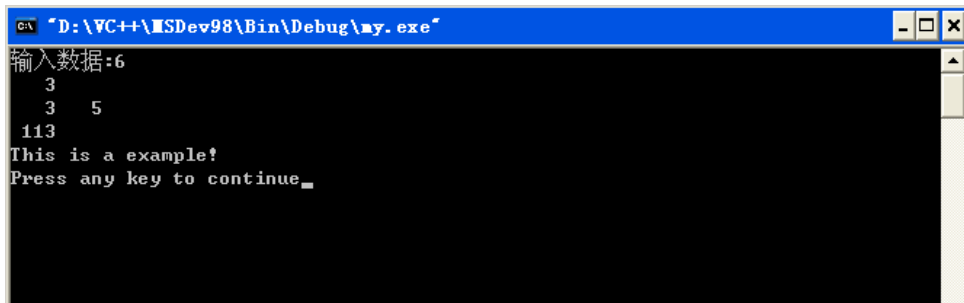
```
01  #include "f.c"                                /*包含 f.c 文件*/
02  #include "c.h"                                /*包含 c.h 文件*/
02  void main()
03  {
04      H x=3,y=5,r,s;                            /*调用宏定义来定义变量*/
05      K c[]="This is a example!";
06      P("输入数据:");                          /*宏定义 P 表示 printf*/
07      scanf(M,&r);                               /*从键盘获取数据至变量 r 中*/
08      s=PI*r*r;                                  /*将 PI*r*r 的值赋给变量 s*/
09      P(D,x);                                    /*调用宏定义输出结果*/
10      P(D1,x,y);
11      P(D,s);
12      P(S,c);
13  }
```

【代码分析】 本例将用户自定义文件包含进来，详细代码分析如下：

- 第 1、2 行，通过 `include` 命令将 `f.c` 和 `c.h` 两个文件包含至 `cfile.c`，因此可直接使用两个文件中的内容。
- 第 4 行，用宏名 `H` 定义变量，相当于整型变量。



【运行结果】该程序的执行结果如图 14-6 所示。



```

D:\VC++\MSDev98\Bin\Debug\my.exe
输入数据:6
3
3 5
113
This is a example!
Press any key to continue_

```

图 14-6 范例 14.6 结果图



注意:

(1) 一个#include 命令只能包含一个文件,若要包含多个文件则应使用多次#include 命令。

(2) 文件包含可以嵌套调用,即包含的文件中包含了另外一个文件。

例如:

```
#include "file1.c"
#include "file2.c"
```

其中 file2.c 中包含 file1.c 中的内容。



14.3 条件编译

在 C 语言中,除了注释部分都可以进行编译运行。但若有时想对满足条件的某一部分进行编译,即通过判断条件的真值来控制其是否进行编译,这称为条件编译。条件编译包含以下三种形式:

(1) 第一种形式:

```
#ifdef 用户标识符
    程序 1
#else
    程序 2
#endif
```

或

```
#ifdef 用户标识符
    程序
#endif
```

上述形式作用为判断用户标识符是否被宏定义,若进行了宏定义则执行程序 1,否则执行程序 2,其中#ifdef 和#endif 分别为起始和结束标志。后一种形式只对宏定义进行判断,若满足

则执行程序，否则不执行后面的程序。

【范例 14.7】 通过下面的例子，简单了解条件编译。

分析：条件编译与 if 语句类似，都是对条件进行判断，只是条件编译对宏定义进行判断而已。

范例 14.7 代码实现

```

01  #include <stdio.h>
02  #include <math.h>                /*包含 math.h 头文件*/
03  #define L(x,y)  x*x-y*y          /*宏定义*/
04  #define H(x,y)  x*y
05  #define TEST
06  void main()
07  {
08      float a,b,m,n;
09      printf("输入两个数:");
10      scanf("%f%f",&a,&b);
11      #ifdef TEST                    /*若宏定义 TEST 存在*/
12          printf("a=%f\nb=%f\n",a,b); /*输出变量 a 和 b 的值*/
13          printf("m=%f\n",L(a,b));    /*输出 L(a,b) 的值*/
14          printf("n=%f\n",H(a,b));    /*输出 H(a,b) 的值*/
15      #else                          /*若宏定义 TEST 不存在*/
16          m=a+L(a,b);                 /*将 a+L(a,b) 赋值给 m*/
17          n=b+H(a,b);                 /*将 b+H(a,b) 赋值给 n*/
18          printf("m=%f\n",m);         /*输出 m 的值*/
19          printf("n=%f\n",n);         /*输出 n 的值*/
20      #endif
21  }

```

【代码分析】 本例为条件编译简单范例，详细代码分析如下：

- 第 11~20 行，对宏进行条件编译。因为 TEST 有宏定义，因此执行第 12~14 行代码，第 15~19 行不执行。

【运行结果】 该程序的执行结果如图 14-7 所示。

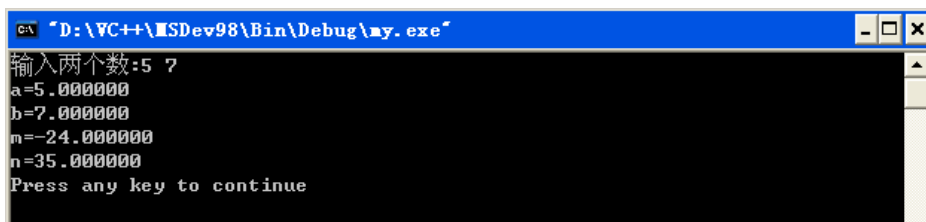


图 14-7 范例 14.7 结果图

(2) 第二种形式：

```

#ifdef 用户标识符
程序 1

```




```
#else
    程序 2
#endif
```

这种形式与第一种形式类似，只是它将 `#ifdef` 变为了 `#ifndef`，与第一种形式正好相反，若用户标识符未进行宏定义则执行程序 1，否则执行程序 2。

(3) 第三种形式：

```
#if 表达式
    程序 1
#else
    程序 2
#endif
```

这种形式与 `if-else` 语句形式类似，当表达式为真时，执行程序 1，当表达式为假时，执行程序 2，可以实现选择性的执行程序。

【范例 14.8】从键盘输入一串字符，利用条件编译将其全部变为小写字母。

分析：利用条件编译进行判断，若为大写字母则减去 32 变化为小写字母，若为小写字母则保持不变。

范例 14.8 代码实现

```
01  #include <stdio.h>
02  #define L 1                                /*宏定义 L 表示 1*/
03  void main()
04  {
05      char s[]="C is a language!",ch;        /*定义字符数组 C 及字符变量 ch*/
06      int i=0;
07      while((ch=s[i])!=''\0')                /*while 循环*/
08      {
09          i++;
10          #if L                                /*L 有宏定义并且其值为 1*/
11              if(ch>='A'&&ch<='Z')            /*若字符 ch 为大写字母*/
12                  ch=ch+32;                  /*转化为小写字母*/
13          #else                                /*否则将小写字母转化为大写字母*/
14              if(ch>='a'&&ch<='z')
15                  ch=ch-32;
16          #endif                                /*结束宏定义判断*/
17          printf("%c",ch);                    /*输出变量 ch 中的字符*/
18      }
19      printf("\n");
20  }
```

【代码分析】本例通过条件编译改变字母大小写，详细代码分析如下：

- 第 5 行，定义了一个字符数组 C，并对其进行初始化。
- 第 10~16 行为条件编译过程。若字符为大写字母则减去 32 转化为小写字母，否则不变。



【运行结果】该程序的执行结果如图 14-8 所示。

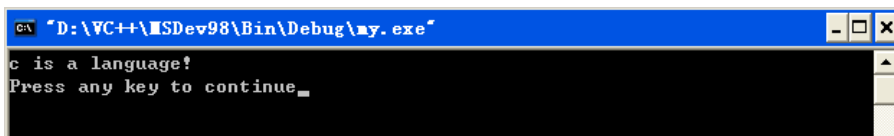


图 14-8 范例 14.8 结果图



14.4 不同存储类型的变量

在 C 语言中,从作用域来看变量可以分为局部变量和全局变量,从生存期看变量可以分为静态变量和动态变量。

静态变量在程序运行的过程中有着固定的存储空间,动态变量在程序运行过程中没有固定的存储空间,它是根据需求动态地分配存储空间的。C 语言中有多种不同存储方式的变量,在本节中将重点介绍这些类型的变量。

14.4.1 自动类型变量

自动型变量指函数内的变量,其通常放在动态存储区,用关键字 `auto` 进行类型声明。例如:

```
void f()  
{  
    auto int x,y;  
}
```

在函数 `f()` 内声明两个自动变量 `x` 和 `y`。当调用函数时,系统将会动态地为变量 `x` 和 `y` 分配其存储空间,函数调用完后,则 `x` 和 `y` 的存储空间会被释放。一般 `auto` 关键词可以省略,因此:

```
int x,y;  
auto int x,y;
```

上述两个语句是等价的。

由于自动变量是在函数内部定义的,因此也被称做内部变量,其作用域为函数内部,生存期为函数调用至函数调用结束。

14.4.2 静态变量

静态变量用关键字 `static` 进行声明,其定义形式如下:

```
static 变量名;
```

例如:

```
static int x,y;
```



由浅入深学 C 语言——基础、进阶与必做 430 题

静态变量在程序运行过程中，会被存放在固定的存储空间，其生存期为定义处至整个程序结束。如果静态变量被定义在函数中，则函数调用完后静态变量的值不会消失也不会改变。当再次调用函数时，静态变量的值仍然为上次调用完函数时的值。

静态变量只能在编译时进行一次初始化，程序运行过程中不会对静态变量进行初始化。

【范例 14.9】通过下面的例子，简单了解静态变量的概念。

分析：静态变量从程序开始至程序结束会一直存在，在函数中也不会消失。

范例 14.9 代码实现

```
01  #include <stdio.h>
02  int f()                      /*自定义函数 f()*/
03  {
04      int x=2;
05      static int y=8;          /*定义静态变量 y*/
06      y++;                     /*y 的值加上 1*/
07      return(x+y);            /*返回 x+y 的值*/
08  }
09  void main()
10  {
11      int i;
12      for(i=0;i<7;i++)         /*通过 for 循环多次调用 f()函数*/
13          printf("%d\n",f());
14  }
```

【代码分析】本例为静态变量简单范例，详细代码分析如下：

- 第 2~8 行，自定义函数 f()，其中 x 为自动变量，y 为静态变量。
- 第 12、13 行，利用 for 循环重复执行 f()函数 7 次。y 的值由 8 最终变化为 15，变量 x 由于为自动变量，因此其值始终为 2。

【运行结果】该程序的执行结果如图 14-9 所示。

图 14-9 范例 14.9 结果图



注意：

(1) 静态变量在程序执行过程中会始终存在，自动变量只在函数内部有效，函数调用完后其存储即被释放。

(2) 静态变量在程序执行过程中，只进行一次初始化，重新执行函数时不会重新初始化，而是保留上次的值进行计算。



(3) 静态变量若为进行初始化, 则系统自动为其赋值为 0。自动变量是根据需求动态分配的, 因此其值是不确定的。

(4) 当需要保留上一次结果时, 才使用静态变量, 一般不使用静态变量。因为使用静态变量会使程序可读性降低。

【范例 14.10】利用静态变量计算 1~10 的阶乘。

分析: 静态变量在函数中值不会改变, 因为计算下个数的阶乘可以通过静态变量的值和该数字进行计算得出。

范例 14.10 代码实现

```
01  #include <stdio.h>
02  long f(int x)                /*自定义函数 f()*/
03  {
04      static long i=1;          /*定义静态长整型变量 i*/
05      i=i*x;                    /*将 i*x 赋值给变量 i*/
06      return i;                /*返回 i 值*/
07  }
08  void main()
09  {
10      int j;
11      for(j=1;j<=10;j++)        /*通过 for 循环和 f() 函数计算 1~10 的阶乘*/
12          printf("%d!=%ld\n",j,f(j));
13  }
```

【代码分析】本例利用静态变量计算阶乘, 详细代码分析如下:

- 第 4 行, 定义了一个静态变量 `i`, 并赋值为 1。
- 第 11、12 行, 通过 `for` 循环和调用 `f()` 函数, 输出 1~10 的阶乘值。

【运行结果】该程序的执行结果如图 14-10 所示。

```
D:\VC++\MSDev98\Bin\Debug\my.exe
1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
Press any key to continue_
```

图 14-10 范例 14.10 结果图

14.4.3 寄存变量

在 C 语言中, 当要经常使用某些数据时, 可将这些数据保存在 CPU 寄存器中, 这样可以快速地存取数据, 提高程序执行效率。



寄存变量是指存放在 CPU 中的变量，它用关键字 `register` 定义。当定义一个寄存变量后，该变量不能再声明为其他类型。

【范例 14.11】 利用寄存变量计算 1~10 的阶乘。

分析：寄存器变量存放 CPU 中，可以方便快捷地进行访问，寄存变量为自动变量，函数调用结束后变量存储空间会被释放。

范例 14.11 代码实现

```
01  #include <stdio.h>
02  long f(int x)                                /*自定义函数 f()*/
03  {
04      register int i;                          /*定义寄存器变量 i*/
05      register int s=1;                        /*定义寄存器变量 s 并进行初始化*/
06      for(i=1;i<=x;i++)                       /*通过 for 循环计算其阶乘*/
07          s=s*i;
08      return(s);                              /*返回 s 值*/
09  }
10  void main()
11  {
12      register int j;                          /*定义寄存器变量 j*/
13      for(j=1;j<=10;j++)                     /*通过 for 循环输出 1~10 的阶乘*/
14          printf("%d!=%ld\n",j,f(j));
15  }
```

【代码分析】 本例利用寄存变量计算阶乘，详细代码分析如下：

- 第 4、5 行，定义了两个寄存变量 `i` 和 `s`。

【运行结果】 该程序的执行结果如图 14-11 所示。

```
D:\VC++\MSDev98\Bin\Debug\my.exe
1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
Press any key to continue
```

图 14-11 范例 14.11 结果图



注意：

- (1) 只有函数内部变量和形参可以声明为寄存变量，其他的变量不能声明为寄存变量。
- (2) 寄存变量不能定义过多，因为 CPU 存储空间不是很大。
- (3) 一个变量只能声明为自动变量、静态变量和寄存变量中的一种。



14.4.4 外部变量

外部变量为全局变量，指定义在函数外部的变量。外部变量的生存期为从定义处至文件结尾，其保存在固定的内存空间中。在 C 语言中，可以通过 `extern` 声明变量，该变量为全局变量，在整个程序中可以被使用。

一般全局变量只有在定义在函数之前，才能被使用，而使用 `extern` 关键字定义变量声明可以在程序的任意位置。例如：

```
void main()
{
    extern a,b;
    int c;
    c=a*b;
    printf("%d",c);
}
int a=5,b=6;
```

使用 `extern` 关键字定义 `a,b` 为全局变量，其中 `a` 和 `b` 的初始化在调用之后，但由于有 `extern` 关键字声明，因此不会出错。

上述程序等价于以下程序：

```
int a=5,b=6;
void main()
{
    int c;
    c=a*b;
    printf("%d",c);
}
```

通过外部变量还可以引用另一个文件中的变量，但不能与当前文件中的变量名冲突。

例如现有以下两个文件：

(1) `f1.c` 文件中的内容如下：

```
extern s;
void main()
{
    int a=3,b=4,c;
    c=a*b*f(a,b);
    printf("%d",c);
}
```

(2) `f2.c` 文件中的内容如下：

```
int s=5;
int f(int m,int n)
{
```



```
int k;  
k=m+n;  
return k;  
}
```

f1.c 文件通过 `extern` 声明变量 `s`，因此可以使用 f2.c 文件中的变量 `s`。



14.5 程序应用举例

【范例 14.12】编写一个程序，计算半径 1~10 的圆的面积、体积及面积和体积的累加和。

分析：计算一个圆的面积和体积可以分别写成一个函数，然后通过调用相应的函数计算圆的面积和体积。同时可以定义两个全局变量，分别计算面积和体积的累加和。

范例 14.12 代码实现

```
01  #include <stdio.h>  
02  #define PI 3.1415926  
03  float s=0;                                /*定义全局变量 s*/  
04  float v=0;                                /*定义全局变量 v*/  
05  float f1(int x)                           /*自定义函数 f1()*/  
06  {  
07      float y;  
08      y=PI*x*x;                             /*计算圆的面积*/  
09      s=s+y;  
10      return(y);                           /*返回 y 值*/  
11  }  
12  float f2(int x)                           /*自定义函数 f2()*/  
13  {  
14      float y;  
15      y=4*PI*x*x*x/3.0;                    /*计算圆的体积*/  
16      v=v+y;  
17      return(y);                           /*返回 y 值*/  
18  }  
19  void main()  
20  {  
21      int i;  
22      for(i=1;i<=10;i++)                   /*输出 i 的值*/  
23          printf("%8d",i);  
24      for(i=1;i<=10;i++)                   /*输出不同半径的圆的面积*/  
25          printf("%8.1f",f1(i));  
26      for(i=1;i<=10;i++)                   /*输出不同半径的圆的体积*/  
27          printf("%8.1f",f2(i));  
28      printf("面积和为:%8.2f\n",s);  
29      printf("体积和为:%8.2f\n",v);  
30  }
```

【代码分析】本例通过全局变量和自动变量计算圆的面积和体积，详细代码分析如下：

- 第 5~11 行，自定义函数 f1()，该函数的功能为计算圆的面积及其累加和。
- 第 22~29 行，利用 for 循环分别输出半径 1 到 10 的圆的面积和体积及其累加和。

【运行结果】该程序的执行结果如图 14-12 所示。

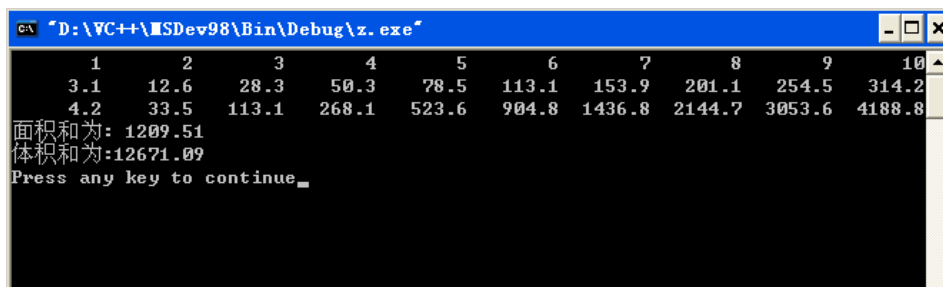


图 14-12 范例 14.12 结果图

【范例 14.13】编写程序，计算长方形的面积。

分析：长方形的面积为长乘以高，可以将长和高定义为全局变量，求长方形的面积定义为一个函数，再在主函数中调用该函数求解。

范例 14.13 代码实现

```

01  #include <stdio.h>
02  float f()                      /*自定义函数 f()*/
03  {
04      extern float x,y;          /*外部声明变量 x 和 y*/
05      float s;                  /*定义浮点型变量 s*/
06      s=x*y;                    /*将 x*y 的值赋给变量 s*/
07      return s;                 /*返回 s 值*/
08  }
09  void main()
10  {
11      printf("面积为%.2f\n",f());
12  }
13  float x=5,y=6.3;
    
```

【代码分析】本例为外部变量应用范例，详细代码分析如下：

- 第 2~8 行，自定义函数 f()，用来计算长方形的面积并返回，其中 x 和 y 为外部变量。
- 第 13 行，定义函数外部变量 x 和 y 并对其进行初始化。

【运行结果】该程序的执行结果如图 14-13 所示。

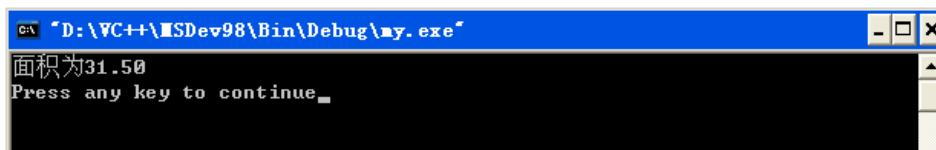


图 14-13 范例 14.13 结果图



14.6 小结

本章重点讲解了编译预处理的过程，其中包括宏定义、文件包含和条件编译等。合理使用宏可以使 C 语言程序简洁清晰，读者应认真学习和掌握。



14.7 习题

一、选择题

1. 以下程序的运行结果为 ()。

```
#define H(a,b,c) a*b-c
void main()
{
    int x=2,y=3,z=5;
    printf("%d",H(x,y+5,z));
}
```

A. 1

B. 6

C. 11

D. 10

【提示】调用宏定义 H(a,b,c)，计算其结果并输出至屏幕。

2. 以下程序的运行结果为 ()。

```
#define F(x) x*x
void main()
{
    int a=5;
    printf("%d",F(a+1));
}
```

A. 36

B. 11

C. 10

D. 9

【提示】通过宏定义 F(a+1)，即 $a+1*a+1$ 的值。

3. 设有以下程序：

```
#define x 2
#define y x+1
#define n 2*y+1
void main()
{
    int i;
    for(i=0;i<n;i++)
        printf("%d",i);
}
```



则其中 for 循环执行的次数为 ()。

- A. 6 B. 5 C. 9 D. 10

【提示】本题 n 值为嵌套宏定义，值为 $2 \times 3 + 1$ ，因此 for 循环的执行次数为 6 次。

4. 以下为 f1.h 头文件中的内容：

```
#define x 5
#define y 6
```

则以下程序的运行结果为 ()。

```
#include "f1.h"
#include "stdio.h"
#define m x*2
void main()
{
    int s;
    s=m+y;
    printf("%d",s);
}
```

- A. 16 B. 15 C. 14 D. 17

【提示】本题包含 f1.h 头文件，同时利用里面的宏定义进行相应的运算和输出。

5. 以下程序的运行结果为 ()。

```
#define M(a) a*(a+1)
void main()
{
    int x=1,y=3;
    printf("%d",M(x+y));
}
```

- A. 16 B. 20 C. 15 D. 21

【提示】上述程序中 M(x+y) 代入表达式中得 $x+y*(x+y+1)$ 。

6. 现有以下程序，则其运行结果为 ()。

```
#define a 5
#define b a*a-1
#include <stdio.h>
void main()
{
    int x,y;
    x=a;
    y=b;
    printf("%d,%d",--x,y);
}
```

- A. 4, 24 B. 4, 20 C. 5, 24 D. 5, 20



由浅入深学 C 语言——基础、进阶与必做 430 题

【提示】本题利用宏定义进行简单的运算，再输出至屏幕。

7. 以下程序的运行结果为 ()。

```
#define M(a,b) a>b?a:b
void main()
{
    int x=5,y=4,t;
    t=M(x+3,y+5);
    printf("%d",t);
}
```

- A. 5 B. 6 C. 9 D. 4

【提示】本题将 $M(x+3,y+5)$ 的值赋给变量 t ，然后输出至屏幕。

8. 以下程序运行结果为 ()。

```
#define x 4
#define y x*x
void main()
{
    int a=5;
    int b=y;
    printf("%d,%d",a,b);
}
```

- A. 5, 4 B. 5, 16 C. 4, 5 D. 4, 16

【提示】上述程序利用宏定义进行运算，输出变量 a 和 b 至屏幕。

9. 以下程序运行结果为 ()。

```
#define f(x) x*x
#define s(x) (x)*(x)
void main()
{
    int m=5;
    int a,b;
    a=f(m-1);
    b=s(m-1);
    printf("%d,%d",a,b);
}
```

- A. -1 -1 B. -1 25 C. -1 16 D. 16 16

【提示】上述程序通过宏定义 $f(x)$ 和 $s(x)$ 进行运算，输出结果至屏幕。

10. 以下程序运行结果为 ()。

```
#define s(x) x*x
#define f(x,y) 2*s(x)-y
void main()
{
```



```
int a=5,b=4;
printf("%d,%d",s(a),f(a,b));
}
```

A. 25, 16

B. 25, 46

C. 24, 16

D. 25, 36

【提示】变量 a 的初始值为 5，b 的初始值为 4，计算宏 s(a)和 f(a,b)的值再输出至屏幕。

11. 以下程序运行结果为 ()。

```
#define x 5
#define f(x) x*x-3
void main()
{
    int y;
    y=f(x);
    printf("%d ",y);
    #undef x
    #define x 6
    y=f(x);
    printf("%d",y);
}
```

A. 22 33

B. 25 36

C. 20 30

D. 以上结果都不正确

【提示】本题通过宏命令及条件编译进行相应的运算和输出。

12. 以下程序运行结果为 ()。

```
#define a 4
#define b a-2*a-1
void main()
{
    int x,y;
    x=a-2;
    y=b-1;
    printf("%d,%d",x,y);
}
```

A. 2 6

B. 2 -6

C. 2 5

D. 2 -5

【提示】宏定义中 a 的值为 4，b 的值为-5，因此 x 和 y 的值为 2 和-6。

二、填空题

1. 以下程序的运行结果为_____。

```
#define P 4
#define M(x) P*x*x
void main()
{
    int x=3,y=4;
```



```
printf("%d",M(x+y));  
}
```

【提示】上述程序中 $M(x+y)$ 代入表达式得 $4*x+y*x+y$ 。

2. 以下程序运行结果为_____。

```
#define x 4  
#define y 5  
void main()  
{  
    int a=x,b=y;  
    #ifdef x  
        printf("%d",a);  
    #else  
        printf("%d",b);  
    #endif  
}
```

【提示】本题中 x 有宏定义，因此输出变量 a 的值，即为 x 的值。

3. 以下程序运行结果为_____。

```
#include <stdio.h>  
#define a 4  
#define b a*a+1  
void main()  
{  
    int x=a,y=b;  
    printf("%d,%d",x,y);  
}
```

【提示】上述程序中 a 的值为 4， b 的值为 17，再将变量 a 和 b 分别赋给变量 x 和 y 。

4. 以下程序的运行结果为_____。

```
#define P(x) printf("%d",x)  
void main()  
{  
    int x=5,y=2;  
    P(x);  
    P(y);  
}
```

【提示】上述程序调用宏定义计算 $P(x)$ 和 $P(y)$ 。

5. 以下程序的运行结果为_____。

```
#define M(x) x*x  
void main()  
{  
    int x=5;
```



```
printf("%d",M(x+4));  
}
```

【提示】上述程序调用宏命令计算 $x+4*x+x$ ，即 $5+4*5+4$ 的值。

6. 以下程序的执行结果为_____。

```
#define S  
void main()  
{  
    int x=5,y=4,z;  
    z=x-y;  
    #ifdef S  
    printf("x=%d",x);  
    #else  
    printf("y=%d",y);  
    #endif  
    printf("z=%d",z);  
}
```

【提示】上述程序利用条件编译及宏命令进行相应的输出。

7. 以下程序的运行结果为_____。

```
#define x 5  
#define y x-2*x+2  
void main()  
{  
    int a=x;  
    int b=y+3;  
    printf("%d,%d",a,b);  
}
```

【提示】上述程序中 x 值为 5， y 的值为 -3，因此 a 和 b 的值为 5 和 0。

8. 以下程序运行结果为_____。

```
#define x 10  
void main()  
{  
    int y=x*x-5;  
    printf("%d ",y);  
    #undef x;  
    #define x 5  
    y=x*x-1;  
    printf("%d",y);  
}
```

【提示】上述程序开始宏定义 x 值为 10，后面取消了宏定义 x ，将 x 值重新定义为 5，参与算术运算后输出。



9. 以下程序运行结果为_____。

```
#define x 6
#define f(x,y) y-x*x
void main()
{
    int a=20;
    int b=f(x,a);
    printf("%d,%d",a,b);
}
```

【提示】上述程序利用带参数的宏参与运算得出结果，再输出至屏幕。

10. 以下程序运行结果为_____。

```
#define min(x,y) x<y?x:y
void main()
{
    int a=5,b=4;
    int c;
    c=min(a-1,b+2);
    printf("%d",c);
}
```

【提示】上述程序通过宏定义 min，计算表达式 $a-1 < b+2$? $a-1 < b+2$ 的值输出至屏幕。

11. 以下程序运行结果为_____。

```
#define P printf
void main()
{
    int x=5,y=4;
    P("%d,%d",x,y);
    P("%d",x+y);
}
```

【提示】本题通过宏定义 P 输出变量 x 和 y 及 x+y 的值至屏幕。

12. 以下程序运行结果为_____。

```
#define x 3
#define y 4
#define z x+y-5
void main()
{
    int a=x;
    int b=y-1;
    int c=z+3;
    printf("%d,%d,%d",a,b,c);
}
```



【提示】本题宏定义 x 值为 3, y 值为 4, z 值为 2, 因此 a, b, c 的值为 3,3,5。

13. 以下程序运行结果为_____。

```
#define letter 1
void main()
{
    char s[]="acjaAHNldne",ch;
    int i=0;
    while((ch=s[i])!='\0')
    {
        i++;
        # if letter
        if(ch>='a'&&ch<='z')
            ch=ch-32;
        #else
        if(ch>='A'&&ch<='Z')
            ch=ch+32;
        #endif
        printf("%c",ch);
    }
}
```

【提示】本题通过条件编译将字符数组 s 中的小写字母转化为大写字母, 然后输出字符串至屏幕。

三、编程题

1. 编写函数求出 1~100 中的素数。

【提示】素数是不能被 1 和其本身外的数整除的数, 因此可设置一个标志, 利用 for 循环进行判断。若能被整除, 则标志值为 0, 否则标志为 1, 最后输出标志为 1 的数即为素数。

【核心代码】

```
for(i=1;i<=100;i++)
{
    flag=1;
    for(j=2;j<=i/2;j++)
        if(i%j==0)
            flag=0;
    if(flag!=0)
        printf("%d ",i)
}
```

2. 利用带参数的宏, 从三个数中找出最大值。

【提示】在使用宏定义时应注意指令替换只是用字符替换, 最终执行效果是一样的。

【核心代码】

```
#define max(x,y) x>y?x:y
scanf("%d%d%d",a,b,c);
```




```
d=max(a,b);  
m=max(d,c)  
printf("%d",m);
```

3. 从键盘输入一个数，利用宏求出其平方值及其平方根。

【提示】利用宏定义求一个数的平方值及其平方根，可以分别用宏名表示对应的字符串。

【核心代码】

```
#define H(x) x*x  
#define S(x) sqrt(x)  
#include <math.h>  
printf("%d,%d",H(x),S(x));
```

4. 从键盘输入一个圆锥的半径和高度，通过宏定义求出圆锥体积。

【提示】圆锥面积的计算公式为 $V=3.14 \times r \times r \times h/3$ ，将从键盘输入的半径和高度套用宏通过公式即可计算出圆锥的面积。

【核心代码】

```
#define V(r,h) 3.14*r*r*h  
S=V(r,h);  
printf("%d",S);
```

第 4 篇 C 语言开发案例

第 15 章 软件设计基础

随着社会的发展, 计算机软件在人们的生活中起着越来越大的作用, 其应用领域非常广泛。软件是一种逻辑的产品, 它能按照人们的要求进行相应的操作, 可以很简单地实现人与计算机交互的功能。软件是计算机工作的首要基础, 不可缺少。在本章中将重点讲解软件的概念、特点及其设计过程, 读者通过本章应了解软件的概念、模型及其设计过程。

本章主要涉及的知识有:

- 软件的特点;
- 软件发展历程;
- 操作系统;
- 数据库操作;
- 软件模型;
- 软件开发过程。



15.1 程序设计语言基础

当前流行的语言主要包括 C、C++、BASIC、Java、PHP、Pascal 等语言, 下面将介绍这几种语言的发展历程及其特点。

1. C 语言

C 语言发展非常迅速, 并且成为当今最流行的语言之一, 主要是由于其强大的功能。很多著名的系统软件都是由 C 编写的, 例如 DBASE IV、PC-DOS、WORSTAR 等。

C 语言具有以下优点:

(1) 简单、灵活方便。(2) 运算符范围广泛。(3) 数据结构类型丰富。(4) 限制不大, 有很大的自由空间。(5) 可直接访问地址, 对计算机硬件操作。(6) 可执行效率高。(7) 适用范围很大, 可移植性好。(8) 绘图功能强大。(9) 适用范围大, 适合于多种系统。

2. C++语言

C++是建立在 C 语言基础上的, 因此取名为 C++。C++相比 C 语言增加了很多功能, 其中包括类、对象、抽象、虚函数、重载等概念。C++是一种面向对象的语言, 其具有以下特点:

(1) C++支持数据封装。在 C++中, 通常将要封装的数据定义成一个类, 而对象即为类的一个实例, 与之相比 C 语言的数据封装性较差。

(2) C++类中包括私有、共有和保护三种不同权限的成员。私有成员只有类中定义的函数能访问, 类外的函数不能访问。保护成员类内及该类派生类中的函数可以访问, 其他的函数不能访问该成员。公共成员类内及类外的函数都可以访问该成员。



(3) C++通过消息来处理对象，每个对象根据其接收到的消息进行相应的操作，来响应该消息。处理对象是通过一系列的方式实现的，这些方法是在函数内部定义的。

(4) C++的友元可以访问其他类中的私有成员，包括友元函数和友元类。这是 C++的一个重要特性，它打破了类的封装性。

(5) C++允许函数和运算符的重载，可以根据实际参数的不同调用相应的函数或运算符，来实现不同的功能。

(6) C++中包含单继承和多继承。一个类可以有一个子类，或有多个子类，这与现实生活的实际情况很相符。

(7) C++支持动态联编及虚函数，可以简洁高效地实现所需功能。

3. BASIC 语言

BASIC 语言是 1964 年由美国的计算机科学家 G.Kemeny 和 Thomas E.kurtz 提出的，它是建立在 FORTRAN 的基础上创造的。它是 Beginner's All-purpose Symbolic Instruction Code（初学者通用符号指令代码）的缩写，当时仅有 17 条语句、12 个函数和 3 个命令。由于其易学性，成为初学者学习计算机程序设计的首选语言。

BASIC 语言具有以下特点：

(1) 语法简单。BASIC 语言基本的控制语句只有 17 种，而且很容易理解和掌握。

(2) 是一种人机交互的语言。人们在使用它编写程序进行编译运行修改时，会提示出现的错误，即实现了人机交互的功能。

(3) BASIC 语言应用十分广泛。

BASIC 语言编译时采用解释器，即对程序逐句翻译成机器代码。翻译完一句后就执行该句，即边翻译边执行程序，这样虽然耗费时间，但可以减少内存的消耗。

4. Java 语言

Java 语言是一种跨越平台的，适合多种机型的面向对象的编程语言。

Java 语言具有以下特性：

(1) 简单性：Java 语言与 C 语言和 C++语言类似，使程序员很容易学习 Java。同时 Java 舍弃了 C++中那些复杂、容易出错的特性，如操作符重载、多继承等。另外 Java 语言中不包含指针的概念，即不使用指针，自动对废料进行回收。

(2) 面向对象：面向对象是指将现实世界中的实体都看做一个对象，对象之间通过消息进行通信。同时，现实世界中一个实体都可以包含于某一类事物中，因此一个对象为一类事物的特殊实例。

Java 语言包含类、接口和继承等功能，但只支持类之间的单继承，类的接口之间可以实现多继承。Java 语言支持动态绑定，而 C++ 语言只对虚函数进行动态绑定。总而言之，Java 语言是一个完全的面向对象程序设计语言。

(3) 分布式：Java 语言可应用于 Internet 的开发，在 Java 的接口中有一个网络应用的编程接口，其可用于网络程序的开发。

(4) 健壮性：Java 语言中的类型机制、异常处理等功能都十分强大，同时舍弃了指针，大大提高了程序的可执行性。另一方面，Java 语言有完善的安全体系，可以防止恶意代码的袭击，具有很强的保护功能。



(5) 中立性: Java 程序被编译为后缀名是.class 的文件后, 可以在装有 JDK 环境的任何操作系统中运行, 不会受操作系统的影响。

(6) 可移植性: Java 语言的可移植性与其中立性有关, 不受操作系统的影响, 因此可以移植到任意的操作系统中。

(7) 解释型: 在 Java 程序的运行过程中, Java 程序被逐行解释运行。

(8) 高性能: Java 语言运行速度很快, 具有很高的性能。

(9) 多线程: Java 语言支持多线程, 即同一时间有多个线程同时运行。另外 Java 语言中还包括了多线程之间的同步机制。

5. PHP 语言

PHP 是一种结合了 C、Java、Perl 及本身创新的语言。PHP 可以快速创建动态网页, 其执行效率相比 CGI 要高很多。PHP 具有强大的功能, 大部分所需功能其都能实现, 适合于多种操作系统及数据库。

PHP 语言具有以下特点:

- (1) 源代码开放, 从网上可以直接看到 PHP 源代码。
- (2) PHP 是完全免费的, 不像 ASP 等要收取一定的费用。
- (3) PHP 学习起来十分简单, 而且使用也很快捷, 适合于初学者。
- (4) PHP 以脚本语言为主, 与 C、C++ 不同。
- (5) PHP 开发程序占据系统资源少。
- (6) PHP 的面向对象特性有着很大的改善, 完全可以用来开发大型应用程序。

6. Pascal 语言

Pascal 语言是一种通用的高级语言, 是由瑞士的一名教授提出的。Pascal 语言功能强大, 使用方法简单, 其具有以下特点:

(1) Pascal 语言是一种结构化语言。它提供了三种基本结构语句, 可以快捷方便地编写出程序。在程序设计过程中, 可以不用 goto 语句, 提高了程序的可读性及正确性。

(2) 结构类型丰富。Pascal 语言包含整型、实型、字符型、枚举型等结构类型, 丰富的结构类型使得 Pascal 语言可以用来方便地解决复杂的问题。

(3) 应用性广。Pascal 语言可以适用于数值或非数值领域, 其在计算机中应用非常广泛。同时 Pascal 语言还可以应用在计算机绘图领域。

(4) 书写格式自由。Pascal 语言与 FORTRAN 相反, 对书写的格式没有严格的要求。因此用 Pascal 语言编写程序时显得很轻松。



15.2 操作系统基础知识

计算机系统分为很多种, 其中包括 Windows、Linux、UNIX 等。不同的操作系统有着不同的特点及其区别, 在本节中将重点讲述这些操作系统的知识。



15.2.1 操作系统分类

操作系统一般可以分为三种，分别为批处理系统、分时系统和实时系统。随着计算机的发展，后来又出现了个人操作系统、网络操作系统和分布式系统等操作系统。

1. 批处理系统

(1) 工作方式

批处理系统的工作方式：用户将作业交给系统管理员，系统管理员收到作业后并不立即运行，而是继续收取作业，直到组成一批作业再将作业输入到计算机中。

(2) 特点及其分类

批处理操作系统是成批处理的，根据系统的复杂度，可以分为简单批处理系统和多道批处理系统。

(3) 系统思想

批处理系统是在早期出现的，因此也被称为早期批处理系统。其思想为：操作员将作业分成一批批作业，然后将一批作业输入到计算机中并执行。

(4) 指令

指令包括一般指令和特殊指令。其中特殊指令包括输入/输出等指令，只有监控程序才能执行。用户执行一般程序，不能执行特殊指令。

(5) 系统调用

系统调用过程分为三步，如下所示。

① 当系统调用时，处理器通过中断机制或异常处理机制将控制流程转交给监控程序，同时转化为特权模式。

② 监控程序执行要运行的代码，实现特定的功能。

③ 执行完成后，从监控程序返回以前的模式，控制权转交给用户程序。

2. 分时系统

分时系统出现在批处理系统之后，它弥补了批处理系统不能与用户实现交互的缺陷。

(1) 工作方式

分时系统工作方式：一台主机连接若干个终端，用户通过终端向系统发出命令，系统接收命令后处理该命令返回至该终端上，从而实现与用户交互的功能。

(2) 系统思想

分时系统将 CPU 时间划分为若干块，并以每块为单位，逐个为用户服务。

(3) 系统特点

- 多路性：一台计算机可以被多个用户使用。
- 交互性：用户可根据系统反馈结果提出请求。
- 独占性：用户感觉计算机就在为自己服务一样，感觉不到为其他人服务。
- 及时性：分时操作系统能够及时快速地对用户命令做出回应。

一般操作系统结合分时与批处理系统两者的特点，例如 UNIX 系统。



3. 实时操作系统

实时操作系统是指计算机在规定时间内响应及处理用户的请求，并控制设备与任务的协调一致。实时操作系统目标为在规定时间内，响应用户的请求，具有高度可靠性。

实时操作系统可以分为两类：硬实时操作系统和软实时操作系统。硬实时操作系统对事件的处理及其时间有着严格的规定，软实时操作系统对事件的处理及其时间有着一定的规定，但要求不是很严格，不会引发灾难性结果。

实时操作系统具有以下特点：

(1) 实时管理

实时操作系统可以对任务进行实时处理。实时任务可以分为定时任务和延时任务两种，其中定时任务是指在规定时间内运行，延时任务指允许延迟运行。

(2) 防护功能

实时操作系统中若出现超过容载的现象，会快速分析其中最重要的实时任务，抛弃不重要的任务，以保证重要任务的成功执行。

(3) 高度可靠性

实时操作系统一般具有高度的可靠性，因为若系统出现故障，则可能会出现灾难性的后果。

4. 个人计算机系统

个人操作系统是一种单独的用户操作系统，主要是个人使用，其功能强大，价格非常便宜，一般都能满足个人用户的需求。个人操作系统在一段时间内只为一个用户服务，同时能实现人机交互的功能，使用起来也很简便。

5. 网络系统

网络操作是指为网络应用配置的系统。网络操作系统是应用于计算机网络等领域的操作系统，其可将网络中的计算机连接起来，实现信息的相互通信及共享。

6. 分布式系统

分布式系统是指将计算机通过网络连接在一起从而提高运算能力及实现数据的共享。分布式系统具有以下特点：

(1) 分布式系统为统一的系统，所有用户使用的都为同一个系统。

(2) 资源用户之间可以实现共享。

(3) 通过分布式操作系统，用户可以清楚本地主机与非本地主机的区别。

(4) 分布式系统中各个用户权限平等，没有特权用户。

分布式系统将多台机器连接在一起，成本很低，但其运算能力很高，同时具有很高的可靠性。

15.2.2 DOS 简介

DOS 是指磁盘操作系统，英文名为 Disk Operating System，是一种个人操作系统，它在 1981 年至 1995 年在市场中占据着重要的位置。

DOS 包括 MS-DOS、PC-DOS、DOR-DOS、FREEDOS 等，其中 MS-DOS 应用最为广泛。



DOS 是一种面向磁盘的系统，通过 DOS 可以很简单地实现人与计算机的交流。自从 DOS 出现后，人们就不必去深入研究计算机的硬件及其复杂枯燥的指令。DOS 不仅可以执行某些指令，还可以管理系统资源，即对其进行调整，使系统有条不紊的工作。

DOS 一直被视为最早最原始的操作系统，今天 DOS 系统已发展为具有各种各样工具及程序的操作系统。DOS 是由 Tim Paterson 在 1980 年 8 月提出的，那时 CP/M 操作系统在市场上占据重要地位。第一个 DOS 系统是由 CP/M 转化过来的，其保留了文件模块功能。开始 DOS 系统并没有多少人知道，后来随着 Paterson 加入 Microsoft 及 DOS 系统的改善，DOS 系统渐渐占据了整个市场。

DOS 系统基本由一个引导程序及三个模块组成，分别为输入输出模块、DOS 程序模块和命令解释模块。当今最流行的 DOS 系统为 MS-DOS 系统，下面讲解其常用的一些命令。

1. 磁盘的操作

`fdisk /mbr`: 重新建立主引导记录。

`format /q`: 快速格式化磁盘。

`format /u`: 不可恢复。

`format /s`: 创建一个 MS-DOS 引导盘。

2. 目录的操作

`DIR[目录名][/S]`: 查找目录中的子目录。

`DIR[目录名][/W]`: 显示目录中的文件名。

`DIR[目录名][/P]`: 分页。

`DIR[目录名][/A]`: 显示目录中的隐藏文件。

例如:

`DIR f.exe /s` 为查找磁盘中 f.exe 文件并指出其位置。

`CD [目录名]`: 可以进入某一特定目录。

`cd ..`: 表示进入文件的上一个目录。

`cd \`: 表示返回文件的根目录。

例如:

`cd c:\Program Files` 表示进入 C 盘的 Program Files 文件夹。

`MKDIR [目录名]`: 创建名为目录名的文件夹。

例如:

`MKDIR FILE`: 表示创建了一个名为 FILE 的文件夹。

3. 文件的操作

`rmdir [目录名]`: 删除目录及其里面的文件。

例如:

`mdir [c:\download]`: 表示删除 C 盘下的 download 目录文件。

`del [目录名][/f]`: 删除目录下的只读文件。

`del [目录名][/s]`: 删除目录下的所有文件。

例如:



`del c:\down /f`: 表示删除 C 盘下的 down 目录中的只读文件。

`copy [目录名][目标目录名]`: 复制目录下的文件到目标目录中。

例如:

`copy c:\download d:\file`: 表示复制 C 盘下的 download 目录中文件到 d 盘的 file 目录中。

15.2.3 Windows Server 2000 与 Windows Server 2003

Microsoft Windows Server 2000 是由微软公司于 1999 年开发出的 32 位操作系统。Windows Server 2000 于 1999 年 12 月首度上市, 它是一个可中断化、有着良好图形界面的商业化操作系统。

Windows Server 2000 有 4 个版本, 分别为 Windows 2000 professional (专业版)、Windows 2000 Server (服务器版)、Windows 2000 Advanced Server (高级服务器版)、Windows 2000 Datacenter Server (数据中心服务器版)。

Windows 2000 系统具有以下功能。

(1) 远程计算功能: Windows 2000 中集成了新型的通信技术, 包括远程连接、拨号、虚拟网络等, 商业伙伴可以在任何时间及地点相互访问信息及应用程序。

(2) 安全性: Windows 2000 是一个具有高度安全性的平台, 它会对用户重要信息进行访问的控制。同时具有多种管理工具, 使得 Windows 2000 的安全性大大提高。

(3) 基于 Web 工作: Windows 2000 是以 Web 方式工作的。Windows 2000 中包含大量的 Internet 计算, 适合于多种服务操作系统。同时 Windows 2000 中包含大量的管理支持工具, 例如系统管理界面等, 从而实现系统有条不紊的管理工作。

(4) 简单易用: Windows 2000 系统使用起来非常简单, 其界面友好易懂, 极大地方便了用户的使用, 用户一般很容易就能上手。

但目前大部分软件都不支持 Windows 2000 操作系统, 同时其不支持 ADSL 拨号, 因此 Windows 2000 系统用户并不是很多。

Windows Server 2003 是微软公司最新推出的服务器操作系统, 与 Windows Server 2000 比较, 该版本做了很大的改进, 尤其是在脚本和命令工具上, 其中包括 Active Directory (活动目录)、Group Policy (组策略) 等。

Windows Server 2003 也包含 4 个版本, 分别为 Windows Server 2003 Web 版、Windows Server 2003 标准版、Windows Server 2003 企业版、Windows Server 2003 中心数据版。

2005 年 3 月 30 日微软公司推出了 Windows Server 2003 的首个服务包, 即 Service Pack 1。其中包括了安全设置、及时更新、Windows 防火墙、媒体播放器等功能。2007 年 Windows Server 2003 Service Pack 2 上市, 其中增添了管理控制面板、开发服务、系统配置等功能。

Windows Server 2003 安全配置包含以下 5 项:

(1) 关闭没用的端口。(2) 关闭不需要的服务, 打开其策略。(3) 关闭没有使用的共享连接。(4) 对磁盘设置权限。(5) 防火墙默认设置。

15.2.4 Linux

Linux 是一款免费使用的操作系统, 其主要用于 CPU 为 x86 系列的计算机上。该系统是由



成千上万个程序员编写出来的，与 UNIX 系统类似。

Linux 系统最早是由名为 Linus Torvalds 的计算机爱好者提出的，当时他还是芬兰赫尔辛基大学的一名学生。他想设计一个操作系统，用来取代 Minix 系统，并且具有 UNIX 系统的所有功能，因此开始了 Linux 系统的设计。

1991 年 4 月，Linus Torvalds 由于对 Minix 系统不满意，因此他自己设计了一个系统核心 Linux 0.01，并将其放在网上，宣布可以免费下载使用并希望大家一起来完善该系统。这时该程序只有 10000 行代码，随着全世界程序员的修改，Linux 系统渐渐完善了，最后终于开发出来一个自由并且具有 UNIX 系统所有功能的 Linux 系统。

Linux 系统具有以下特点。

(1) 成本低廉

Linux 具有高度可设性，同时其成本较低廉，常被应用于嵌入式系统中。另外，有些网络防火墙及路由都使用 Linux。

(2) 应用广泛

随着 Linux 系统的发展，使用 Linux 系统的用户越来越多。据统计，Linux 系统已成为世上最快速的操作系统之一。虽然 Linux 代码是公布的，但人们不必要研究该代码，只要掌握 Linux 的简单应用及安装就可以了。

(3) 使用灵活

Linux 系统具有高效性及灵活性，它具有 UNIX 系统的所有特性，如多任务、多个用户的功能。Linux 系统不仅具有 UNIX 系统的所有功能，同时还包含文本编辑器、语言编辑器等软件。它具有 Windows 图形界面，人们可以直接通过屏幕对计算机进行操作。

Linux 系统是一款自由的系统，它不收取任何的费用，同时其代码在网上是公开的。人们可以根据自己的需要来修改代码，可以无约束的传播。另一方面，Linux 系统具有 UNIX 系统的所有功能，想学习 UNIX 系统的用户可以从 Linux 系统中学习。

随着 Linux 系统的完善，相信 Linux 系统发展前景会越来越好，其功能也会越来越强大。



15.3 数据库基础知识

随着社会的发展，越来越多的数据需要进行保存，因此数据库的应用范围越来越大。人们通过数据库系统可以对数据进行统一的管理、修改及保存，可以很方便地对数据进行访问和操作。同时使用数据库系统也使得数据的安全性大大提高，在本节中将讲述数据库系统基础知识及其应用。

15.3.1 时下流行的数据管理系统简介

数据库管理系统是一种对数据进行操作和管理的系统，可用于建立、访问和维护数据，其英文名为 DataBase Management System，简称为 DBMS。

数据库管理系统大致可分为以下 6 个部分。

- 翻译模式：数据库逻辑结构、存储结构等被保存在内部中，同时数据库的查找、修改



等操作及管理维护都是根据数据库模式来执行的。

- 程序编译：将包含访问数据库操作的程序编译成可执行的目标代码程序。
- 指令查询：包含查询语言，可以对指令进行查询。
- 数据存取：数据在外围设备（磁盘等）上有着特定的存储方法。
- 事务管理：包含事务管理及运行日志，以及数据检测、备份、恢复等功能。
- 数据维护：包含多种工具，如数据的完整性检测、数据备份、数据重组等工具。

数据库管理系统已经被广泛应用在商业的各个领域中，随着数据库系统技术的广泛，其应用领域将进一步扩展。根据对象的不同，数据库管理系统包括应用层、语言翻译层、数据存取层、操作系统等不同的层次结构。

当前流行的数据库管理系统包括 Oracle、Sybase、Informix、Microsoft SQL Server、Microsoft Access、Visual FoxPro 等，它们在市场上都占据着一定的地位。下面讲解这几种常用的数据库管理系统。

1. Oracle

Oracle 是最早的数据库管理系统，该系统应用十分广泛同时功能也很强大。Oracle 不仅具有完善的数据处理功能，而且是一个分布式系统，支持多种分布式应用，如 Internet 应用等。其界面友善，功能完善齐全，具有可开放性、可移植性等特性，具有面向对象的功能。现今 Oracle 已更新至 Oracle 11g 版本。

2. Sybase

Sybase 是由美国 Sybase 公司于 1987 年推出的关系型数据库管理系统，可以运行在 Windows 和 UNIX 等操作系统中。Sybase 中包括丰富的编程接口及类库，允许多个数据库之间进行操作，具有良好的数据安全性。

Sybase 数据库具有以下特点。

（1）基于客户/服务器体系结构：一般数据库都基于主/从结构，所有的操作都在一台机器中，用户通过终端发送命令查看结果。而在客户/服务器体系结构中，所有应用被分配到多台机器中，可以实现资源的共享，降低了机器的负载。（2）开放数据库：Sybase 数据库管理系统公开了程序接口 DB-LIB 的代码，可以改写其接口代码，因此其具有很强的开放性。（3）高性能：Sybase 数据库具有很高的性能，对数据访问及存取效率很高，可以快速访问及修改数据库系统中的数据。

3. Microsoft SQL Server

Microsoft SQL Server 为 Microsoft、Sybase 及 Ashton-Tate 三家公司共同推出的关系数据库管理系统。SQL Server 2000 是 Microsoft 公司推出的数据库管理系统，其继承了以前 SQL Server 的优点，同时增添了多种功能，具有使用方便、集成度高等特点。

Microsoft SQL Server 具有以下特点。

（1）可信性：Microsoft SQL Server 具有很高的安全性、集成性及扩展性。Microsoft SQL Server 对数据可以进行审查及加密处理，大大提高了数据的可靠性及安全性。（2）高效性：Microsoft SQL Server 系统具有很高的性能，可以降低开发和管理数据所需的成本及时间，大大方便了数据库的设计。（3）智能型：Microsoft SQL Server 为一个智能全面的平台，可以根据用



户的需求发送相应的信息。

4. Microsoft Access

Microsoft Access 是由微软发布的关联数据库管理系统。Microsoft Access 能够存取多种类型的资料，其中包括 Access、SQL Server、Oracle 等数据。该系统的界面友好，使用起来十分方便，同时也能满足大部分用户的需求。

5. Visual FoxPro

Visual FoxPro 是由 Microsoft 公司推出的数据库管理系统，使用它可以方便快捷地开发数据库。Visual FoxPro 是在 xBASE 上发展形成的，如今 VFP6 是世界上最流行的功能强大、性能优异的软件之一。

Visual FoxPro 具有以下特点：

(1) Visual FoxPro 是面向对象编程的，并且提供多种可视化工具。(2) 可以直接结合表的字段和控件。

15.3.2 SQL 语言简介

SQL 语言是指结构化查询语言，其英文名为 Structured Query Language。SQL 语言是由 IBM 公司的圣约瑟研究实验室在 1981 年提出的，它是在 SQUARE 语言的基础上发展而来的。SQL 语言功能强大，使用起来简单方便，现如今流行的 Oracle、Sybase、SQL Server 等数据库系统都支持用 SQL 语言作为其查询语言。

SQL 语言包括数据查询、数据操纵、数据定义、数据控制 4 个部分。20 世纪 70 年代中期，IBM 公司在研制数据库系统过程中提出了 SQL 语言，在 1976 年 11 月首度上市。1979 年，ORACLE 公司首次将 SQL 商业化，后来 IBM 公司也在其数据库系统中实现了 SQL。1986 年 10 月，SQL 语言被美国采用作为数据库系统标准化语言，后被认定为国际标准语言。

SQL 具有如此广泛的应用，与其强大的功能密不可分。SQL 具有以下特点。

(1) SQL 为非过程化语言

SQL 是一种非过程化的语言，它每次只处理一个记录，对数据进行操作。SQL 允许用户对记录集进行操作，而不是对单个记录进行操作。SQL 语言作为一个集合输入计算机中，所有结果作为一个集合从计算机中输出。SQL 对用户数据存取方法没有很严格的要求，这使得存储过程简单很多，同时更关注于结果，而不用担心存储过程。

(2) SQL 是一种统一的语言

SQL 可被应用于多种模型，其中包括系统管理员、数据管理员等。SQL 语言简单易学，简单的指令很快就能学会，复杂的指令几天基本上也可以掌握。其语言中包括多种命令，其中包括查询、插入和删除记录、建立和修改数据、检测数据库等，因此 SQL 作为一门语言，功能十分强大。

(3) SQL 是所有数据库的公共语言

Oracle、Sybase、SQL Server 等数据库系统都支持 SQL 语言，因此用 SQL 语言编写的程序完全可以移植到另一个系统中。



15.4 软件工程

要进行软件的设计,就必须要先学习软件工程。学习软件工程,可以以较低的成本开发出易理解、易维护、符合用户要求的软件。

软件工程是指采用工程的概念及其方法来,用最好的技术及方法来开发软件。软件工程指导人们如何进行软件的开发,其中包括计划与估算、需求分析、数据结构、总体结构设计、算法设计、编写程序及测试维护等。在本节中将重点讲解软件的概念及其设计方法。

15.4.1 软件概念

软件是一种计算机逻辑产品,它在计算机中可以被使用,并且人们通过软件可以实现想要执行的操作,如删除、修改数据等。软件是计算机的基础,没有软件计算机将不能工作。例如现在人们使用的计算机系统都为系统软件,若没有系统软件,计算机将失去工作的平台,不能进行工作。

软件与计算机中的硬件相互依存,其包含以下三方面的内容:

(1) 运行过程中能提供相应的程序来响应用户的操作。(2) 含有相应的数据结构。(3) 描述软件使用过程及其方法的文档。

随着计算机的快速发展,计算机与人的沟通越来越困难,因此软件的作用越来越大,其规模也越来越大。软件具有以下特点:

(1) 软件具有抽象性,它没有具有的物理实体。人们不能看到软件的外貌,但可以通过测试、分析知道软件的功能、性能及其他特性。(2) 软件是人们通过实践将知识与技术结合的产品。软件研制过程成本要远远高于其生产成本,研制过程包括分析、思考、设计、测试等步骤。(3) 软件不像硬件,不会随着使用次数及时间出现磨损和老化的现象。但软件会出现故障,而且其维护比硬件的维护要困难的多。(4) 软件的开发受计算机硬件的影响,软件是运行在计算机硬件的基础上的。(5) 软件开发需要投入大量的人力、物力,同时软件的开发费用也很庞大。(6) 软件设计开发是一个复杂漫长的过程,必须要对软件的开发进行合理的管理和分析。

15.4.2 软件分类

人们在平时使用计算机的过程中,经常会碰到各种各样的软件。这些软件可以从不同的角度分为不同的类型,下面讲解如何对软件进行分类。

1. 通过软件功能分类

(1) 系统软件

系统软件是指与计算机硬件联系紧密的软件,如操作系统、数据库管理系统等。系统软件常常与计算机硬件打交道,处理复杂的进程及数据。系统软件在计算机中不可缺少,它是计算机正常工作的基础。

(2) 支持软件

支持软件是指帮助用户开发软件及控制进程的工具软件。它可以分为以下几种类型。



- 一般型：文本编辑程序、文件程序。
- 需求型：问题描述器、检测器。
- 设计型：图形报、绘图程序、设计程序。
- 实现型：编辑程序、连接程序、执行程序。
- 测试型：分析程序、模拟程序、测试程序。
- 管理型：评审器、检验程序、管理程序。

(3) 应用软件

应用软件是指满足用户某种需求的软件，如数据处理软件、嵌入软件和识别软件等。

2. 通过软件工作方式分类

通过软件不同的工作方式可以把软件分为以下几种类型。

(1) 分时型软件：分时型软件允许同一时间内多个用户使用计算机，轮流处理每个用户，可使用户感觉不到其他用户同时在使用计算机。(2) 交互型软件：交互型软件可以实现人与计算机的通信，它接收用户的请求作出相应的响应，同时对时间没有严格的规定，大大提高了软件的灵活性。(3) 批处理型软件：批处理型软件不对一个作业进行处理，而是对每批作业进行逐个地处理。

3. 通过软件规模分类

通过软件规模的大小可以把软件分为以下 6 种类型。

(1) 微型软件：微型软件是指代码行不超过 500 行仅供个人使用的软件。通常该类软件没有严格的要求，设计和测试也不是很严格。(2) 小型软件：小型软件是指代码行在 2000 行内的软件，其开发过程有一定标准要求，书写及测试也有一定的要求。(3) 中型软件：中型软件是指代码行在 50000 行内的软件。在中型软件中，程序员和用户之间开始存在联系，以协调开发软件。(4) 大型软件：大型软件是指代码在 50000~100000 行内的程序，由 5~10 个人完成。对于该类软件，其设计过程审查要求比较严格。由于规模较大，在设计过程中往往会出现不可预料的错误。(5) 甚大型软件：甚大型软件是指具有 100 万行的代码程序，它由 100~1000 人完成。这种类型项目会划分成若干个子项目，其中每个项目都为大型的软件。每个子项目间都有相互通信的接口。(6) 超大型软件：超大型软件是指代码行在 100 万行~1000 万行内的程序，由 2000~5000 人完成。这类软件很少，往往运用于军事、导弹等方面。

4. 通过软件服务对象分类

根据软件完成后服务的对象可以分为以下两类。

(1) 定制类型软件：定制类型软件一般是由某些用户委托开发的，它是由一个或多个软件开发团队开发设计的。(2) 产品类型软件：产品类型软件开发完后直接面向市场，为所有用户服务，可供任何人使用。

15.4.3 软件开发流程

软件开发流程主要包括三个阶段，分别为计划阶段、开发设计阶段、运行及维护阶段，同时这三个阶段又可以分为以下阶段。



(1) 计划阶段

计划阶段包括软件设计和需求分析阶段。

- 软件设计：这一阶段主要从当前系统出发，定义软件的性质及其功能等，提出对软件的设计方案及成本估计等。
- 需求分析：这是一个很重要的阶段。通过与用户交流了解用户所需的功能，根据用户需求再进行相应软件的编写。

(2) 开发设计阶段

开发设计包含设计、编写程序、测试三个阶段。首先对数据的结构进行设计，建立相应的数据结构，然后对数据结构编写代码操作，最后对其进行测试，看能否运行成功。

(3) 运行及维护阶段

运行及维护阶段主要实现对软件进行维护。根据软件的运行状况，进行相应的维护。

15.4.4 软件开发模型

任何一个软件开发之前，不管其规模大小，都必须要有相应的模型，而且要根据软件的性质、功能选用合适的模型，一个错误的模型可能会导致整个软件开发的失败。

1. 瀑布模型

瀑布模型是由 B.M.Boehm 提出的软件设计基础模型。其思想为将功能与设计分开从而实现问题的简化。它采用的是结构化方法，将逻辑问题与实际问题分开。瀑布模型中每个阶段如同瀑布一样，逐流而下，上一个阶段完成后下一个阶段才会执行。

通过瀑布模型开发软件可以及时地开发出软件，但该模型不够灵活，无法解决软件需求问题。同时使用瀑布模型前一阶段出现错误，很有可能会影响下一阶段的执行。

2. 原型模型

使用原型模型，则软件开发过程中是按顺时针前进的。软件开发者根据用户的需求快速设计原型，然后对原型进行评估，提出更加精确的要求。最后根据原型，开发设计软件。

原型模型可以完全满足用户的需求，总成本较低，开发时间较短。但开发者有可能会将用户所需的次要部分当成主体部分，从而其原型不切实际。

3. 螺旋模型

螺旋模型是瀑布模型与其他模型的结合体，它克服了瀑布模型的缺点。螺旋模型中沿着螺旋线旋转，其中包括 4 个方面的内容，分别为计划制定、风险分析、工程实施和客户评价。

螺旋模型中第一圈确定初步目标，每旋转一圈软件的版本则更加完善。一般螺旋线至末尾处，便能得到用户所期望的软件。螺旋模型适合作为大型软件的开发模型，相比其他模型来说，其风险性较高，一般用户很难接受。

4. 增量模型

增量模型是融合了顺序模型和原型模型的模型，它采用日程进展的序列。第一个增量往往为软件的核心部分，即实现基本用户要求，随着增量的增多，软件逐渐完善。

增量模型使用灵活，易于人员的分配，具有一定的市场地位。但通过增量模型设计软件时，



开发者和客户必须始终保持交流状态，直至开发完成。

5. 喷泉模型

喷泉模型是一种根据用户需求，以对象为实例的模型，主要用于对象技术软件开发。在该模型中，软件的各个阶段是相互迭代的，没有间隙。如系统某个部分被重复执行多次，则相关的功能在每次迭代中会加入系统中。

喷泉模型与瀑布模型不同，它要分析结束后才开始设计过程，设计过程后再开始编写代码。其优点为节省开发所需时间，提高了开发效率，但由于其开发阶段重叠，因此需要大量的人员来开发软件，成本较高。

6. 智能模型

智能模型是基于知识的模型，它将上述模型与专家模型结合在一起。该模型一般应用于基于规则的系统，通过归纳和推理的机制实现软件的开发，并使维护在系统一级进行。

15.4.5 软件需求分析

需求分析是软件设计过程中的一个重要阶段，它是开发者对软件的“理解、分析和表达”的过程。在这个阶段中，开发者需要准确理解用户的需求，将用户的需要分解并转化为相应的功能。

软件开发之后都是给用户使用的，因此软件必须符合用户的需要，否则即使开发出软件，也将是一个失败品。需求分析主要通过当前系统根据用户要求设计出相应的模型。需求分析包括软件的功能需求分析、性能需求分析、数据需求分析和其他需求分析等。然后根据这些需求，编写程序设计出相应的软件。

需求分析主要根据以下原则进行分析。

(1) 符合客户语言习惯；(2) 了解客户的业务及目标；(3) 编写软件需求报告；(4) 有需求工作的解释说明；(5) 尊重客户意见；(6) 对软件开发提出自己的意见及方案；(7) 描述出产品特性；(8) 尽量使用已有组件；(9) 尽量满足客户需求及其要求；(10) 划分需求的优先级；(11) 评审需求文档。

需求分析的步骤如下：

- (1) 与客户进行交流，获取其所需要求。
- (2) 分析用户需求。
- (3) 编写用户需求文档。
- (4) 与用户进行文档的评审，根据用户要求更改。
- (5) 合理管理需求。

经过需求分析之后，建立相应的模型，就可以编写程序设计相应的软件了。



15.5 小结

在本章介绍了软件的概念及分类，然后讲解了软件模型，其中包括瀑布模型、原型模型、



螺旋模型、智能模型和喷泉模型等。这些概念都是软件设计中必须要掌握的基础知识，读者应认真学习和掌握。



15.6 习题

一、选择题

1. 软件的主体功能是在（ ）阶段确定的。

- A. 分析
- B. 编程
- C. 测试
- D. 维护

【提示】软件设计包括分析、编程设计、测试、维护等阶段，其功能是在分析阶段确定的。

2. 软件模型中最基础的模型为（ ）。

- A. 智能模型
- B. 螺旋模型
- C. 瀑布模型
- D. 喷泉模型

【提示】软件模型中最基础的模型为瀑布模型。

3. 瀑布模型存在的缺点为（ ）。

- A. 与用户交流少
- B. 灵活性低
- C. 与用户交流甚密
- D. 容易变化

【提示】在软件设计过程中通过瀑布模型开发软件可以及时的开发出软件，但该模型不够灵活，无法解决软件需求问题。同时使用瀑布模型前一阶段出现问题，很有可能会影响下一阶段的执行。

4. 在软件设计过程中，用户参与的阶段为（ ）。

- A. 软件分析阶段
- B. 软件编写阶段
- C. 软件维护阶段
- D. 软件开发的整个阶段

【提示】因为软件是设计给用户使用的，在程序开发的整个过程中都应询问客户的意见及需求。

二、简答题

1. 简单阐述软件的特点。
2. 软件可以分为哪些种类？
3. 软件设计过程分为哪几个阶段？
4. 软件模型包含哪些模型？

第 16 章 C 语言程序综合应用

通过前面章节的学习，读者应基本掌握了 C 语言的知识。本章将讲解 C 语言中的一些经典问题，并用 C 语言代码实现。这些问题都是 C 语言中比较经典、困难的问题，其中包括八皇后问题、汉洛塔问题等。经过本章的学习，将有助于开发读者的视野，提高算法分析设计及解决问题的能力，增强读者的思维能力。



16.1 八皇后问题

【范例 16.1】八皇后问题：现有 8×8 的棋盘，要求在其中放入 8 个皇后，可以使任意两个皇后不能吃掉对方。其中皇后可以吃掉同一行、同一列、同一对角线中的其他皇后，试求出其所有可能的解并输出至屏幕。

分析：现有 8 个皇后，可以分别设这 8 个皇后为 X_i ($i=1,2,3\cdots 8$)。因为 $1 \leq X_i \leq 8$ ，因此原问题可以转化为求 $(1, x_1), (2, x_2), (3, x_3), (4, x_4), \cdots (8, x_8)$ 不在同一行、同一列、同一对角线的解的个数。八皇后中不在同一列可用 $X_i \neq X_j$ 来表示，不在主对角线的表达式为 $X_i - i \neq X_j - j$ ，不在负对角线上的表达式为 $X_i + i \neq X_j + j$ 。因此八皇后不在同一对角线上表达式可以合并为 $\text{abs}(x_i - x_j) \neq \text{abs}(i - j)$ 。

范例 16.1 代码实现

算法设计代码 1:

```
01  #include <stdio.h>
02  #include <math.h>                                /*包含 math.h 头文件*/
03  int f(int x[],int n)                               /*自定义函数 f()*/
04  {
05      int i;
06      for(i=1;i<=n-1;i++)
07          if((abs(x[i]-x[n])==abs(i-n))||(x[i]==x[n])) /*比较是否满足条件*/
08              return 0;                                /*若不满足条件返回 0*/
09      return 1;                                        /*若满足条件返回 1*/
10  }
11  void main()
12  {
13      int x[9],i;                                     /*定义整型数组 x 和整型变量 i*/
14      for(x[1]=1;x[1]<=8;x[1]=x[1]+1)                /*第一层 for 循环*/
15          for(x[2]=1;x[2]<=8;x[2]=x[2]+1)            /*第二层 for 循环*/
16      {
```



```
17     if(f(x,2)==0)                                /*调用函数 f()*/
18     continue;
19     for(x[3]=1;x[3]<=8;x[3]=x[3]+1)                /*第 3 层 for 循环*/
20     {
21         if(f(x,3)==0)                                /*调用函数 f()*/
22         continue;
23         for(x[4]=1;x[4]<=8;x[4]=x[4]+1)            /*第 4 层 for 循环*/
24         {
25             if(f(x,4)==0)                            /*调用函数 f()*/
26             continue;
27             for(x[5]=1;x[5]<=8;x[5]=x[5]+1)        /*第 5 层 for 循环*/
28             {
29                 if(f(x,5)==0)                        /*调用函数 f()*/
30                 continue;
31                 for(x[6]=1;x[6]<=8;x[6]=x[6]+1)    /*第 6 层 for 循环*/
32                 {
33                     if(f(x,6)==0)                    /*调用函数 f()*/
34                     continue;
35                     for(x[7]=1;x[7]<=8;x[7]=x[7]+1) /*第 7 层 for 循环*/
36                     {
37                         if(f(x,7)==0)                /*调用函数 f()*/
38                         continue;
39                         for(x[8]=1;x[8]<=8;x[8]=x[8]+1) /*第 8 层 for 循环*/
40                         {
41                             if(f(x,8)==0)            /*调用函数 f()*/
42                             continue;
43                             else
44                             for(i=1;i<=8;i++)        /*输出所有可能的结果*/
45                             {
46                                 printf("%d",x[i]);
47                                 if(i==8)
48                                 printf("\n");
49                                 }}}}}}}}}}
```

【代码分析】本例利用 for 循环穷举求解，详细代码分析如下：

- 第 3~10 行，自定义函数 f()，用来判断两个棋子是否在同一列和同一对角线上。若皇后在同一列或在对角线上则返回值 0，否则返回值 1。
- 第 14~48 行，利用 8 个 for 循环，穷举 8 个皇后所有可能的位置，对每两个皇后进行判断，若在同一列或同一对角线上则返回 for 循环继续执行，否则输出结果至屏幕。

【运行结果】该程序的执行结果如图 16-1 所示。



```
C:\ "D:\VC++\MSDev98\Bin\Debug\my.exe"
1 5 8 6 3 7 2 4
1 6 8 3 7 4 2 5
1 7 4 6 8 2 5 3
1 7 5 8 2 4 6 3
2 4 6 8 3 1 7 5
2 5 7 1 3 8 6 4
2 5 7 4 1 8 6 3
2 6 1 7 4 8 3 5
2 6 8 3 1 4 7 5
2 7 3 6 8 5 1 4
2 7 5 8 1 4 6 3
2 8 6 1 3 5 7 4
3 1 7 5 8 2 4 6
3 5 2 8 1 7 4 6
3 5 2 8 6 4 7 1
3 5 7 1 4 2 8 6
3 5 8 4 1 7 2 6
3 6 2 5 8 1 7 4
3 6 2 7 1 4 8 5
3 6 2 7 5 1 8 4
3 6 4 1 8 5 7 2
```

图 16-1 范例 16.1 结果图

算法设计代码 2:

```
01  #include <stdio.h>
02  #include <math.h>                /*包含 math.h 头文件*/
03  int x[20],n;                      /*定义全局变量整型数组 x 和整型变量 n*/
04  int f(int m)                      /*自定义函数 f()*/
05  {
06      int i;
07      for(i=1;i<=m-1;i++)
08          if(abs(x[i]-x[m])==abs(i-m) || (x[i]==x[m])) /*比较其是否满足条件*/
09              return 0; /*若不满足条件则返回值为 0*/
10      return 1; /*若满足条件返回值为 1*/
11  }
12  void prin(int n)                  /*自定义函数 prin()*/
13  {
14      int i;
15      for(i=1;i<=n;i++)
16      {
17          printf("%d",x[i]); /*输出满足条件的解至屏幕*/
18          if(i==n)
19              printf("\n");
20      }
21  }
22  void back(int n)                  /*自定义函数 back()*/
23  {
24      int m; /*定义整型变量 m*/
25      x[1]=0;
26      m=1;
27      while(m>0)
28      {
29          x[m]=x[m]+1;
```



```
30     while((x[m]<=n)&&f(m)==0)                /*通过 while 循环穷举*/
31         x[m]=x[m]+1;
32         if(x[m]<=n)
33             if(m==n)                          /*输出结果至屏幕*/
34                 prin(n);
35             else                              /*继续求解*/
36             {
37                 m=m+1;
38                 x[m]=0;
39             }
40         else                                  /*返回一个位置*/
41             m=m-1;
42     }
43 }
44 void main()
45 {
46     scanf("%d",&n);                          /*从键盘输入皇后的个数*/
47     back(n);                                  /*调用自定义函数 back()*/
48 }
```

【代码分析】本例利用非递归回溯的算法求解，详细代码分析如下：

- 第 12~21 行，自定义函数 prin()，该函数的功能为输出满足条件的解。
- 第 22~43 行，自定义函数 back()，用回溯的方法求解出所有可能的结果。
- 第 47 行，调用 back()函数求解，其中 n 作为参数传递给函数。

【运行结果】该程序的执行结果如图 16-2 所示。

```
D:\VC++\MSDev98\MyProjects\Debug\hui.exe
8
1 5 8 6 3 7 2 4
1 6 8 3 7 4 2 5
1 7 4 6 8 2 5 3
1 7 5 8 2 4 6 3
2 4 6 8 3 1 7 5
2 5 7 1 3 8 6 4
2 5 7 4 1 8 6 3
2 6 1 7 4 8 3 5
2 6 8 3 1 4 7 5
2 7 3 6 8 5 1 4
2 7 5 8 1 4 6 3
2 8 6 1 3 5 7 4
3 1 7 5 8 2 4 6
3 5 2 8 1 7 4 6
3 5 2 8 6 4 7 1
3 5 7 1 4 2 8 6
3 5 8 4 1 7 2 6
3 6 2 5 8 1 7 4
3 6 2 7 1 4 8 5
3 6 2 7 5 1 8 4
3 6 4 1 8 5 7 2
3 6 4 2 8 5 7 1
3 6 8 1 4 7 5 2
3 6 8 1 5 7 2 4
```

图 16-2 算法 2 运行结果图



算法设计代码 3:

```
01  #include <stdio.h>
02  #include <math.h>
03  int x[20],y[20],z[40],m[40],n,t=0,i,j,k;      /*定义全局变量*/
04  void prin()                                  /*自定义函数 prin()*/
05  {
06      int k;
07      t=t+1;
08      printf("%d ",t);                        /*输出解的个数*/
09      for(k=1;k<=n;k++)                       /*输出满足条件的一组解*/
10          printf("%d ",x[k]);
11      printf("\n");
12  }
13  void f(int i)                                /*自定义函数 f()*/
14  {
15      int j;
16      for(j=1;j<=n;j++)
17          if((y[i]==0)&&z[i+j]==0&&m[i-j+n]==0) /*判断其是否满足条件*/
18          {
19              x[i]=j;
20              y[j]=1;
21              z[i+j]=1;
22              m[i-j+n]=1;
23              if(i<8)                          /*若 i 值小于 8*/
24                  f(i+1);                      /*递归调用函数 f()*/
25              else
26                  prin();                      /*调用函数输出结果*/
27              y[i]=0;                          /*回溯将其元素置 0*/
28              z[i+j]=0;
29              m[i-j+n]=0;
30          }
31  }
32  void main()
33  {
34      int i;
35      scanf("%d",&n);                          /*输入皇后的个数*/
36      for(i=1;i<=n;i++)                       /*初始化棋盘*/
37      {
38          y[i]=0;
39          z[i]=0;
40          z[n+i]=0;
41          m[i]=0;
42          m[n+i]=0;
43      }
44      f(1);                                    /*调用函数 f()*/
45  }
```



【代码分析】本例利用递归的算法求解，详细代码分析如下：

- 第 4~12 行为自定义函数 `prin()`，用来输出求出的解，其中变量 `t` 用来记录解的个数。
- 第 13~31 行，通过递归算法求解八皇后问题。
- 第 23 行，若 `i` 的值小于 8 则重新调用 `f()` 函数，直至 `i` 的值为 8。
- 第 32~45 行，输入皇后的个数并调用函数 `f()` 进行求解。

【运行结果】该程序运行结果与算法设计 2 的运行结果一致。



16.2 汉洛塔问题

从前在古代印度罗门圣庙的僧尚中，有一种名为汉洛塔的游戏。设有柱子 A、B、C，其中柱子 A 上有着若干个大小不等的圆盘，要将柱子 A 上的圆盘完整地移动到 C 柱子上。要求每次只能移动最上面的圆盘，并且可以将柱子 B 作为媒介，但大的圆盘不能在小的圆盘之上。

经过实践证明，汉洛塔问题是可以实现的，但其过程非常复杂。

汉洛塔示意图如图 16-3 所示。

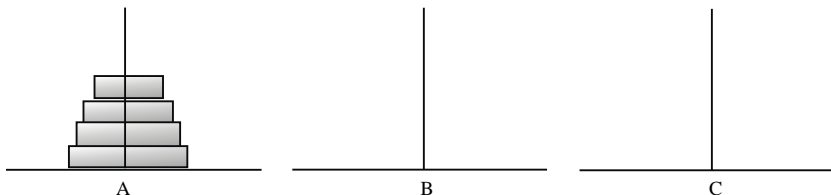


图 16-3 汉洛塔示意图

【范例 16.2】编写一个程序，求解汉洛塔问题。

分析：设 A 柱上有 n 个盘子，因为大盘不能在小盘之上，因此将 A 柱中最大的盘子移动到 C 柱之前，C 柱必须为空。可以先将 A 柱中的 $n-1$ 个盘子先移动到 B 柱上，然后将第 n 个盘子移动到 C 柱上。当 C 柱上已放有最大的盘子时，A 柱为空，B 柱上有 $n-1$ 个盘子。这时可将 B 柱上的 $n-2$ 个盘子移动到 A 柱上，再将 B 柱上的第 $n-1$ 个盘子移动到 C 柱上，如此循环直到最后盘子的个数为 1 为止，直接将盘子移动到 C 盘上，循环结束。

综上所述，其求解步骤如下：

- (1) 通过 C 柱，将 A 柱上的 $n-1$ 个盘子移动到 B 柱上。
- (2) 将 A 柱上剩余的一个盘子移动到 C 柱上。
- (3) 通过 C 柱，将 B 柱上的 $n-2$ 个盘子移动到 A 柱上。
- (4) 将 B 柱上剩余的一个盘子移动到 C 柱上。
- (5) 重复执行第 (1) 步，直到盘子的个数为 1 为止。

范例 16.2 代码实现

```
01  #include <stdio.h>
02  int hanruo(char A,char B,char C,int n)    /*自定义函数 hanruo()*/
03  {
04      if(n==1)                                /*若 n 值为 1*/
```



```
05  {
06    printf("move from %c to %c\n",A,C);          /*输出移动过程*/
07    return 0;
08  }
09  hanruo(A,C,B,n-1);                             /*递归调用 hanruo()函数*/
10  printf("move from %c to %c\n",A,C);          /*输出移动过程*/
11  hanruo(B,A,C,n-1);                             /*递归调用 hanruo()函数*/
12  }
13  void main()
14  {
15    int n;
16    printf("输入盘子的个数:");
17    scanf("%d",&n);                             /*输入盘子的个数至 n 中*/
18    hanruo('A','B','C',n);                       /*调用自定义函数 hanruo()*/
19  }
```

【代码分析】本例利用递归算法求解，详细代码分析如下：

- 第 2~12 行，用户自定义函数 hanruo()。该函数利用递归求解盘子搬运的过程，当 n 等于 1 时，退出递归函数的调用。
- 第 16~18 行，输入盘子的个数并调用函数 hanruo()求解，其中字符 A、B、C、 n 作为参数。

【运行结果】该程序的执行结果如图 16-4 所示。

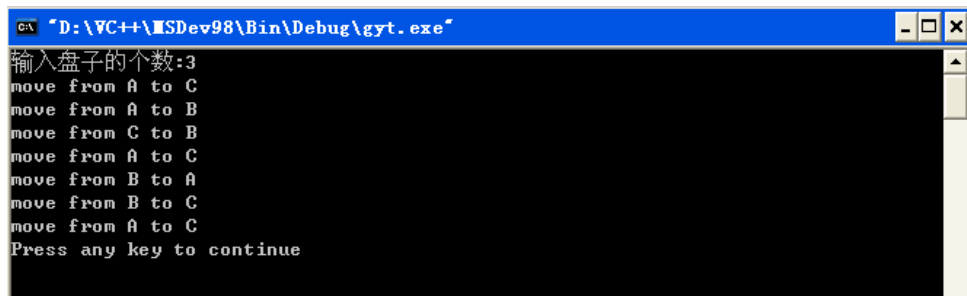


图 16-4 范例 16.2 结果图



16.3 循环赛问题

【范例 16.3】设有 n 个选手要进行乒乓球选拔赛，编写一个程序设计其比赛日程安排表，具有以下要求：

- (1) 每名选手都必须与其他选手比赛一次。
- (2) 每名选手一天之内只能进行一次比赛。
- (3) 所有选手的比赛在 $n-1$ 天内比完。

分析：可以将所有选手分成两半，即 n 个选手的比赛日程由 $n/2$ 个选手的比赛日程决定。



用这种一分为二的方法对选手进行划分直至只剩下两个对手为止，单独设置其比赛日程，然后将这些单独的块最后合并起来即为最终的解。如表 16-1 所示为利用二分法设计出的 8 个选手的比赛日程安排表。

表 16-1 8 名选手比赛安排表

1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

其中第 1 列为每个选手的编号，每一行为每一名选手 7 天内要比赛的对手。

范例 16.3 代码实现

```
01  #include <stdio.h>
02  int x[50][50];                      /*定义二维全局数组 x*/
03  void f()                             /*自定义函数 f()*/
04  {
05      int n,m,i,j,k,z;
06      x[1][1]=1;                      /*将 1 赋值给 x[1][1]*/
07      scanf("%d",&k);                 /*从键盘输入数据至 k 中*/
08      m=1;
09      n=2;
10      for(z=1;z<=k;z++)               /*分组过程*/
11      {
12          for(i=m+1;i<=n;i++)          /*安排比赛日程*/
13          for(j=1;j<=n;j++)
14              x[i][j]=x[i-m][j]+m;
15          for(j=m+1;j<=n;j++)
16          for(i=1;i<=m;i++)
17              x[i][j]=x[i][j-m]+m;
18          for(j=m+1;j<=n;j++)
19          for(i=m+1;i<=n;i++)
20              x[i][j]=x[i][j-m]-m;
21          m=n;
22          n=n*2;
23          for(i=1;i<=n;i++)
24          {
25              printf("\n");
26              for(j=1;j<=n;j++)        /*输出安排好的比赛日程*/
27                  printf("%d ",x[i][j]);
28          }
29      }
```




```
30  }
31  void main()
32  {
33      f();                      /*调用自定义函数 f()*/
34  }
```

【代码分析】本例利用二维递归的方法求解，详细代码分析如下：

- 第 2 行，定义了一个全局二维数组 `x`，用来保存结果。
- 第 3~30 行为自定义函数 `f()`。该函数用递归的方法安排每个选手的比赛日程表，并将结果输出至屏幕。

【运行结果】该程序的执行结果如图 16-5 所示。

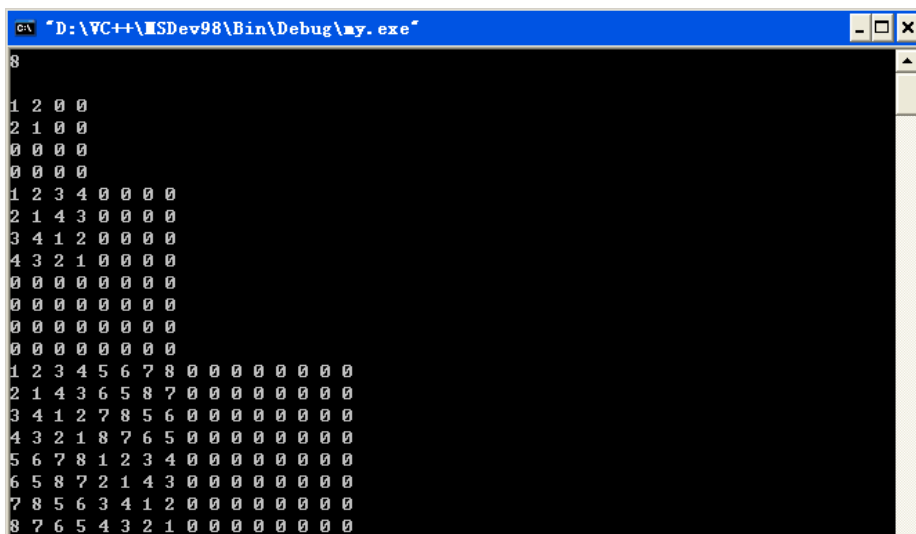


图 16-5 范例 16.3 结果图



16.4 猴子选大王

【范例 16.4】现有 17 个猴子，这些猴子根据一个游戏选举大王。其游戏规则为 17 个猴子站成一圈，然后从 1 到 3 开始计数，数到 3 的猴子淘汰退出游戏，下一个猴子继续从 1 开始计数，直到最后只剩一个猴子，即为大王。

分析：可以将每个猴子抽象成一个编号，并用数组保存每个猴子的状态。然后通过数组模拟其选举过程，直到淘汰 $n-1$ 个猴子选出大王为止。

范例 16.4 代码实现

```
01  #include <stdio.h>
02  void game()                      /*自定义函数 game()*/
03  {
04      int i,n=17,x[17],j;          /*定义全局变量*/
```



```
05  for(i=0;i<n;i++)                /*初始化猴子*/
06  x[i]=1;
07  j=0;                            /*用于计数*/
08  while(n!=1)                    /*直到剩下最后一个猴子*/
09  {
10  for(i=0;i<17;i++)              /*模拟报数过程*/
11  {
12  j=j+x[i];
13  if(j==3)                      /*若数到 3*/
14  {
15  x[i]=0;                      /*该猴子出队列*/
16  j=0;                          /*重新从 0 开始报数*/
17  n=n-1;                        /*猴子总数减 1*/
18  }
19  }
20  }
21  for(i=0;i<17;i++)              /*for 循环枚举找出猴子大王*/
22  if(x[i]==1)
23  printf("猴子大王为第%d 号\n",i);
24  }
25  void main()
26  {
27  game();                        /*调用自定义函数 game()*/
28  }
```

【代码分析】本例利用数组模拟报数求解，详细代码分析如下：

- 第 5、6 行，将数组 x 中的值都赋为 1，即表示都站在圈内。
- 第 8~20 行，模拟报数的整个过程。若数到 3 则该元素值为 0 即表示出列，直到只剩一个猴子为止。
- 第 21~23 行，对数组中的所有元素判断，若其值为 1 则表示为最后一个猴子，即为猴子大王。

【运行结果】该程序的执行结果如图 16-6 所示。

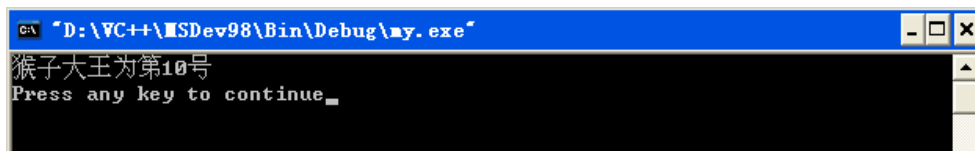


图 16-6 范例 16.4 结果图



16.5 三个数的最小公倍数问题

【范例 16.5】从键盘输入三个数，求其最小公倍数。



由浅入深学 C 语言——基础、进阶与必做 430 题

分析：最小公倍数是指能整除这三个数的公倍数中的最小者，可以利用循环穷举的方法，看是否能整除这三个数。若能整除这三个数，则输出其中的最小的数即为最小公倍数。

范例 16.5 代码实现

算法设计代码 1:

```
01  #include <stdio.h>
02  int max(int a,int b,int c)          /*自定义函数 max()*/
03  {
04      if(a>b&& a>c)                  /*若 a 值最大*/
05          return a;                 /*返回 a 值*/
06      else if(b>a&& b>c)             /*若 b 值最大*/
07          return b;                 /*返回 b 值*/
08      else                           /*否则返回 c 值*/
09          return c;
10  }
11  void main()
12  {
13      int x,y,z,i,j,k;
14      printf("输入三个数:");
15      scanf("%d%d%d",&x,&y,&z);      /*从键盘输入三个数*/
16      k=max(x,y,z);                 /*调用 max() 函数求出其中的大者*/
17      i=1;
18      while(1)                      /*while 循环, 条件值为 1*/
19      {
20          j=k*i;                     /*将 k*i 赋值给变量 j*/
21          if(j%x==0&& j%y==0&& j%z==0) /*若满足条件则退出 while 循环*/
22              break;
23          i++;
24      }
25      printf("最小公倍数为%d\n",j); /*输出最小公倍数*/
26  }
```

【代码分析】本例利用 for 循环逐个比较进行求解，详细代码分析如下：

- 第 2~10 行，自定义函数 max()。该函数用来求解三个数中的最大者，从而求其最小公倍数。
- 第 16 行，调用 max() 函数求出从键盘输入的最大者。
- 第 18~24 行，通过循环改变变量 j 的值，看其是否能整除这三个数。若能整除这三个数，则退出 while 循环。

【运行结果】该程序的执行结果如图 16-7 所示。

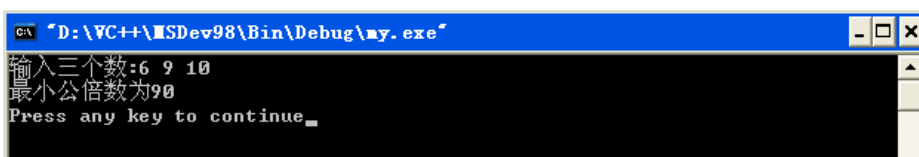


图 16-7 范例 16.5 结果图



算法设计代码 2:

求三个数的最小公倍数也可以通过短除法实现。短除法思想为求三个数的因数，然后求这些因数的乘积。代码如下:

```
01  #include <stdio.h>
02  int max(int a,int b,int c)           /*自定义函数max()*/
03  {
04      if(a>b&&b>c)                     /*若 a 值最大*/
05          return a;                   /*返回 a 值*/
06      else if(b>a&&b>c)                /*若 b 值最大*/
07          return b;                   /*返回 b 值*/
08      else                             /*否则返回 c 值*/
09          return c;
10  }
11  void main()
12  {
13      int x,y,z,t=1,i,j,k;
14      printf("输入 3 个数:");
15      scanf("%d%d%d",&x,&y,&z);        /*从键盘输入三个数*/
16      k=max(x,y,z);                   /*调用函数 max() 求最大值*/
17      for(i=2;i<=k;i++)
18      {
19          j=1;
20          while(j)                     /*while 循环, 条件值为 j*/
21          {
22              j=0;
23              if(x%i==0)                /*若 x 能被 i 整除*/
24              {
25                  x=x/i;                /*将 x 除以 i 的值赋给 x*/
26                  j=1;
27              }
28              if(y%i==0)                /*若 y 能被 i 整除*/
29              {
30                  y=y/i;                /*将 y 除以 i 的值赋给 y*/
31                  j=1;
32              }
33              if(z%i==0)                /*若 z 能被 i 整除*/
34              {
35                  z=z/i;                /*将 z 除以 i 的值赋给 z*/
36                  j=1;
37              }
38              if(j==1)                  /*将 t*i 赋给变量 t*/
39                  t=t*i;
40      }
41      k=max(x,y,z);
```



```
42     printf("最小公倍数为%d\n",t);           /*输出最小公倍数*/
43 }
```

【代码分析】本例利用短除法求最小公倍数，详细代码分析如下：

- 第 17~41 行，利用短除法求三个数的因子，然后求所有因子的乘积，即为最小公倍数，其中 j 作为循环是否进行的标志。

【运行结果】该程序的执行结果与算法 1 一样。

算法设计代码 3:

求三个数的最小公倍数，可以先求两个数的最小公倍数，然后利用函数的嵌套求这三个数的最小公倍数。代码如下：

```
01  #include <stdio.h>
02  int f(int x,int y)           /*自定义函数 f()*/
03  {
04      int a,b,c;
05      a=x;
06      b=y;
07      c=x%y;                   /*将 x 除以 y 的余数赋给变量 c*/
08      while(c!=0)              /*while 循环,执行条件为 c 不等于 0*/
09      {
10          x=y;
11          y=c;
12          c=x%y;               /*将 x 除以 y 的余数赋给 c*/
13      }
14      return a*b/y;            /*返回 a*b/y 的值*/
15  }
16  void main()
17  {
18      int a,b,c,d;
19      printf("输入 3 个数:");
20      scanf("%d%d%d",&a,&b,&c);   /*从键盘输入数据至相应变量中*/
21      d=f(f(a,b),c);           /*嵌套调用函数 f()求解*/
22      printf("最小公约数为%d\n",d); /*输出最小公倍数*/
23  }
```

【代码分析】本例利用函数嵌套功能求三个数的最小公倍数，详细代码分析如下：

- 第 2~15 行，自定义函数 $f()$ ，用来求两个数的最小公倍数。
- 第 21 行，嵌套调用函数 $f()$ 求得三个数的最小公倍数。

【运行结果】该程序的执行结果与算法 1 和 2 结果一样。



16.6 背包问题

背包问题有多种版本，不同版本有着不同的特点，也有着不同的解决方案。



【范例 16.6】 现有 9 件商品，从中选出 3 件使得其重量之和与 600 克之间差值最小，其中每件商品的重量由键盘输入。

分析：由于要选取 3 件物品，可以利用 3 重 for 循环穷举这 3 件物品的所有可能，进行相互之间的比较得出与 600 克最接近的组合。

范例 16.6 代码实现

```
01  #include <stdio.h>
02  #include <math.h>           /*包含 math.h 头文件*/
03  void main()
04  {
05      int best=10000,i,j,k,x,y,z;    /*定义相应变量*/
06      float w,s[10];
07      for(i=1;i<=9;i++)             /*从键盘输入商品重量*/
08          scanf("%f",&s[i]);
09      for(i=1;i<=7;i++)             /*3 重 for 循环*/
10          for(j=i+1;j<=8;j++)
11              for(k=j+1;k<=9;j++)
12              {
13                  w=abs(600-s[i]-s[j]-s[k]);    /*将 600-s[i]-s[j]-s[k]绝对值赋给 w*/
14                  if(w<best)                  /*若 w 值小于 best*/
15                  {
16                      best=w;                  /*将 w 赋给 best*/
17                      x=i;                    /*记录 3 件商品的位置*/
18                      y=j;
19                      z=k;
20                  }
21              }
22      printf("三件商品分别为第%d,%d,%d 号\n",x,y,z);    /*输出商品位置*/
23      printf("总重量为%d\n",best);                    /*输出总重量*/
24  }
```

【代码分析】 本例利用 for 循环穷举所有可能求解，详细代码分析如下：

- 第 7、8，调用 scanf() 函数从键盘输入每一件商品的重量。
- 第 12~21 行，通过 3 重 for 循环逐个与 600 克比较。若小于 best 变量的值，则赋值给变量 best，并将 3 件物品的编号分别赋给 x、y、z。

【运行结果】 该程序的执行结果如图 16-8 所示。

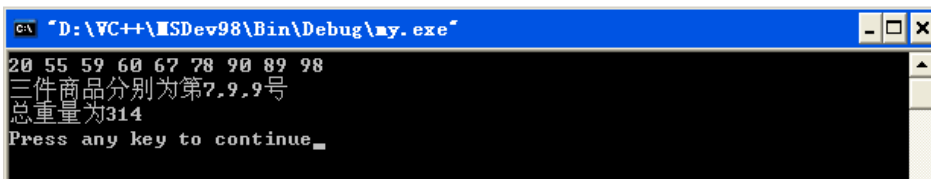


图 16-8 范例 16.6 结果图

【范例 16.7】 现有一个最大容纳质量为 mkg 的背包和 n 件商品，其中每件商品的重量



w_i 和价格 p_i 由键盘输入，编写一个程序使得赚取的利益最大。

分析：若 $w_1 + w_2 + \dots + w_n \leq m$ ，则可以购买所有商品，若 $w_1 + w_2 + \dots + w_n \geq m$ 则需要进行选择性地购买，每次选择价格与质量比例最大的物品即可实现利益的最大化。

范例 16.7 代码实现

```
01  #include <stdio.h>
02  int main()
03  {
04      int m,n,i,j,w[50],p[50],x[50],y[50],z,max;    /*定义相应变量*/
05      printf("输入包的容量及商品的个数:");
06      scanf("%d%d",&m,&n);                          /*输入包的总容量及商品总个数*/
07      for(i=1,z=0;i<=m;i++)
08      {
09          printf("输入每件物品的重量以及价格\n");
10          scanf("%d%d",&w[i],&p[i]);                /*输入每件商品的重量及价格*/
11          x[i]=p[i];
12          z=z+w[i];                                  /*将 z+w[i]赋值给 z*/
13      }
14      if(z<=m)                                        /*若 z 小于或等于 m*/
15      {
16          printf("全部选取\n");
17          return 0;
18      }
19      for(i=1;i<=n;i++)
20      {
21          max=1;
22          for(j=2;j<=n;j++)                          /*for 循环穷举*/
23              if(x[j]/w[j]>x[max]/w[max])              /*若 x[j]/w[j]大于 x[max]/w[max]*/
24                  max=j;                              /*将 j 赋给 max*/
25          x[max]=0;
26          y[i]=max;
27      }
28      for(i=1,z=0;z<m&&i<=n;i++)                    /*统计总重量*/
29          z=z+w[y[i]];
30      if(z!=m)
31          w[y[i-1]]=w[y[i-1]]-(z-m);
32      for(j=1;j<=i-1;j++)                          /*输出选择的商品及其重量*/
33          printf("最终选择为:%d,重量:%d\n",y[j],w[y[j]]);
34  }
```

【代码分析】本例为背包问题范例，详细代码分析如下：

- 第 7~13 行，将每件物品的重量和价格赋值给数组 w 和 p 。
- 第 14~18 行，若所有商品的重量之和小于背包容量，则可以选取所有商品。
- 第 19~27 行，求出价格和重量之比最大的商品，并将其赋给数组 y 。

【运行结果】该程序的执行结果如图 16-9 所示。



```

D:\VC++\MSDev98\Bin\Debug\gyt.exe
输入包的容量及商品的个数:5 20
输入第1件物品的重量以及价格
6 3
输入第2件物品的重量以及价格
2 5
输入第3件物品的重量以及价格
3 8
输入第4件物品的重量以及价格
10 6
输入第5件物品的重量以及价格
7 4
最终选择为:2,重量:2
最终选择为:3,重量:3
最终选择为:1,重量:3
最终选择为:1,重量:3
最终选择为:1,重量:3
Press any key to continue
  
```

图 16-9 范例 16.7 结果图



注意：本例中讲解的为部分背包问题，允许物品拆散。若不允许物品拆散，用贪心算法就不一定能得到最优解了。



16.7 马遍历问题

【范例 16.8】现有 $n \times m$ 个格子的棋盘，马在棋盘上行走，只能走日字。编写程序，求出马从任意位置出发，将所有格子走完并且只能走过一次的所有路径。

分析：棋盘总共有 n 行和 m 列，设马的当前坐标为 (a,b) ，则马走日字之后，坐标的所有可能为 $(a+1,b+2)$ ， $(a+1,b-2)$ ， $(a+2,b+1)$ ， $(a+2,b-1)$ ， $(a-1,b+2)$ ， $(a-1,b-2)$ ， $(a-2,b+1)$ 和 $(a-2,b-1)$ 。通过搜索的方式模拟马在棋盘上走，直到走完棋盘为止。可以用变量 d 存储走过的点数，当 d 值为 $n \times m$ 时表示找到一组解。数组存储马所走的路径，0 表示为经过，1 表示走过。

范例 16.8 代码实现

```

01  #include <stdio.h>
02  int s1[8]={1,2,2,1,-1,-2,-2,-1},s2[8]={2,1,-1,-2,-2,-1,1,2},s[5][4];
                                           /*定义全局数组 s1*/
03  int d,a,b,i,j,sum;                      /*定义全局变量*/
04  void prin()                             /*自定义函数 prin()*/
05  {
06      sum=sum+1;                          /*sum 值加 1*/
07      printf("sum=%d",sum);               /*输出 sum 值*/
08      for(b=1;b<=5;b++)                  /*通过 for 循环输出结果*/
09      {
10          printf("\n");
11          for(a=1;a<=4;a++)
12          printf("%d ",s[b][a]);
  
```




```
13     }
14     }
15     int search(int x,int y)                /*判断是否出界*/
16     {
17         if(x<=5&&y<=4)                    /*若出界返回值为 1*/
18             return 1;
19         return 0;
20     }
21     void f(int a,int b,int d)              /*在棋盘上行走*/
22     {
23         int i,x,y;
24         for(i=1;i<9;i++)
25         {
26             x=a+s1[i];                    /*模拟马走日字*/
27             y=b+s2[i];
28             if(search(x,y)==1)            /*若未出界*/
29             {
30                 s[x][y]=d;
31                 if(d==5*4)                /*走完棋盘输出结果*/
32                     prin();
33                 else                      /*否则递归调用函数 f()*/
34                     f(x,y,d+1);
35             }
36         }
37         s[x][y]=0;                        /*回溯*/
38     }
39     void main()
40     {
41         sum=0;
42         d=1;
43         printf("输入起始点坐标:");
44         scanf("%d%d",&a,&b);              /*从键盘输入起点的坐标*/
45         if(a>5||b>4||a<1||b<1)
46             printf("输入有误\n");
47         for(i=1;i<=5;i++)                /*初始化棋盘*/
48             for(j=1;j<=4;j++)
49                 s[i][j]=0;
50         s[a][b]=1;
51         f(a,b,2);                        /*调用函数 f()*/
52     }
```

【代码分析】本例通过数组模拟马走格子个数为 5×4 的棋盘，详细代码分析如下：

- 第 4~14 行，用户自定义函数 prin()，用来输出马走过的路径。
- 第 15~20 行，用来判断马是否出界，即是否超过棋盘的范围。
- 第 21~38 行，利用回溯算法模拟马走棋盘的过程，并将结果保存至相应的数组中。
- 第 51 行，调用 f()函数求解所有可能的解。



【运行结果】该程序的执行结果如图 16-10 所示。

```
CA "D:\VC++\MSDev98\Bin\Debug\gyt.exe"
输入起始点坐标:2 3

sum=3
0 0 0 0
0 0 1 0
0 0 0 0
0 0 0 3
0 0 0 0
sum=4
0 0 0 0
0 0 1 0
0 0 0 0
0 20 0 4
0 0 0 0
sum=5
0 0 0 0
0 0 1 0
20 0 0 0
0 20 0 5
0 0 0 0
```

图 16-10 范例 16.8 结果图



16.8 流水线作业问题

【范例 16.9】现有 n 个作业和 m 台机器，要求每台机器同一时刻只能做一个作业，每个作业完成需不同的时间，编写一个程序使得 n 个作业在 m 台机器上工作的时间最少。

分析：将 n 个作业分配给 m 台机器中，若 n 值小于 m 则直接分配给机器即可。若 n 值大于 m ，则应从 n 个作业中选取 m 个时间较大的作业分配给 m 台机器，然后依次选取完成作业时间最早的机器继续执行剩余的作业，直到所有作业都完成为止。

范例 16.9 代码实现

```
01  #include<stdio.h>
02  int t[100],d[100];          /*定义全局数组 t 和 d*/
03  struct job                  /*定义结构体数组*/
04  {
05      int h;
06      int t;
07  }job[100],s[100];
08  void sort(int n)            /*自定义函数 sort()*/
09  {
10      int i;
11      for(i=1;i<=n;i++)
12          s[i]=job[i];
13      for(i=1;i<=n;i++)        /*采用冒泡排序法进行排序*/
14          for(int j=i+1;j<=n;j++)
15      {
```



```
16     if(s[i].t<s[j].t)
17     {
18         int k=s[i].t;
19         int w=s[i].h;
20         s[i].t=s[j].t;
21         s[i].h=s[j].h;
22         s[j].t=k;
23         s[j].h=w;
24     }
25 }
26 }
27 void fenpei(int n,int m)                /*自定义函数 fenpei()*/
28 {
29     int i,j,k,min,z;
30     for(i=1;i<=n;i++)                  /*初始化 d[m]*/
31         d[i]=0;
32     for(i=1;i<=m;i++)
33         d[i]=s[i].t;                    /*放入最初的 m 个作业*/
34     for(i=m+1;i<=n;i++)                /*放入剩余的作业*/
35     {
36         for(k=1;k<=m;k++)
37             for(z=k+1;z<=m;z++)        /*从机器中找出工作时间最小的作业*/
38                 if(d[k]<d[z])
39                     j= k ;
40         else
41             j=z;
42         d[j]=d[j]+s[i].t;
43     }                                    /*将剩下的作业放入相应机器中*/
44     for(k=1;k<=n;k++)                  /*求出最短工作时间*/
45         if(d[k]>d[k+1])
46             min=d[k];
47     printf("最短工作时间为:%d\n",min); /*输出最短时间*/
48 }
49 void main()
50 {
51     int n,m,i;
52     printf("输入作业的个数以及机器的台数:");
53     scanf("%d%d",&n,&m);              /*从键盘输入机器台数以及作业个数*/
54     for(i=1;i<=n;i++)
55     {
56         scanf("%d",&t[i]);            /*初始化每个作业的工作时间*/
57         job[i].h=i;
58         job[i].t=t[i];
59     }
60     for(i=1;i<=n;i++)
61     {s[i].h=0;s[i].t=0;}
```



```

62     sort(n);                                /*调用 sort()函数从小到大排序*/
63     fenpei(n,m);                            /*调用 fenpei()函数求解问题*/
64 }

```

【代码分析】本例为流水线作业范例，详细代码分析如下：

- 第 3~7 行，定义了两个结构体数组 job 和 s。
- 第 8~26 行，自定义函数 sort()，用来对数组从小到大进行排序。
- 第 27~48 行，用户自定义函数 fenpei()。该函数的功能为将 n 个作业分配给 m 台机器，先从 n 个作业中选取 m 个时间较大的作业分配给 m 台机器，然后依次选取完成作业时间最早的机器继续执行剩余的作业，直到所有作业都完成。
- 第 62、63 行，调用 sort()和 fenpei()函数求解作业调用问题。

【运行结果】该程序的执行结果如图 16-11 所示。

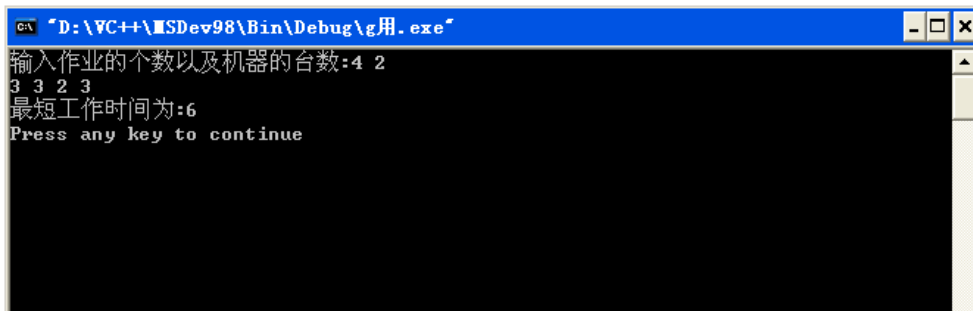


图 16-11 范例 16.9 结果图



16.9 迷宫问题

【范例 16.10】现有一个给定的迷宫，编写一个程序实现从入口走出迷宫。

分析：假设迷宫是由许多格子组成的矩形，其中用 1 表示有墙，0 表示有路。走迷宫即为沿上、下、左、右方向走，直到最后走出迷宫。

范例 16.10 代码实现

```

01     #include <stdio.h>
02     #define m 6                                /*宏定义 m 值为 6*/
03     #define n 9                                /*宏定义 n 值为 9*/
04     void search()                            /*自定义 search()函数*/
05     {
06         int maze[8][11]={1,1,1,1,1,1,1,1,1,1,1,1,    /*初始化迷宫模型*/
07                             1,0,1,0,0,0,1,1,0,0,1,
08                             1,1,0,0,0,1,1,0,1,1,1,
09                             1,0,1,1,0,0,0,0,1,1,1,
10                             1,1,1,0,1,1,1,1,0,1,1,
11                             1,1,1,0,1,0,0,1,0,0,1,

```



```
12         1,0,0,1,1,1,0,1,1,0,1,
13         1,1,1,1,1,1,1,1,1,1,1} ;
14     int m[9][2];
15     int s[54][3];
16     int top=0;
17     int i,j,k,p,f=0,g,h;           /*定义相应变量*/
18     m[1][0]=-1; m[1][1]=0;         /*初始化 m 数组*/
19     m[2][0]=-1; m[2][1]=1;
20     m[3][0]=0; m[3][1]=1;
21     m[4][0]=1; m[4][1]=1;
22     m[5][0]=1; m[5][1]=0;
23     m[6][0]=1; m[6][1]=-1;
24     m[7][0]=0; m[7][1]=-1;
25     m[8][0]=-1;m[8][1]=-1;
26     maze[1][1]=2;
27     s[top][0]=1;
28     s[top][1]=1;
29     s[top][2]=3;
30     ++top;
31     while (top!=0&&f==0)           /*当 top 不等于且 f 不等于 0*/
32     { --top;                       /*top 值减 1*/
33         i=s[top][0];
34         j=s[top][1];
35         k=s[top][2];
36         while(k<=8)                /*求解迷宫过程*/
37         { g=i+m[k][0];
38           h=j+m[k][1];
39           if (g==m&&h==n&&maze[g][h]==0) /*若满足条件*/
40           {
41               for(p=0;p<top;++p)
42               printf("%d,%d,%d\n",s[p][0],s[p][1],s[p][2]); /*输出走过的路径*/
43               printf("%d,%d,%d\n",i,j,k);
44               printf("%d,%d,%d\n",m,n,k);
45               f=1;
46           }
47           if(maze[g][h]==0)          /*若 maze{g}[h]值为 0*/
48           {
49               maze[g][h]=2;
50               s[top][0]=i;
51               s[top][1]=j;
52               s[top][2]=k;
53               ++top;                 /*top 值加 1*/
54               i=g;
55               j=h;
56               k=0;
57           }
```



```

58     k=k+1;
59     }
60     }
61     if(f==0)                                /*若 f 等于 0 表示没有出路*/
62     printf("没有出路\n");
63     }
64     void main()
65     {
66         search();                            /*调用自定义函数 search()*/
67     }

```

【代码分析】本例为迷宫范例，详细代码分析如下：

- 第 6~13 行，利用二维数组 `maze` 初始化迷宫模型，1 表示有墙，0 表示有路。
- 第 31~59，利用循环沿上、下、左、右方向行走，直到找到出口为止。
- 第 66 行，调用自定义函数 `search()` 求解迷宫问题。

【运行结果】该程序的执行结果如图 16-12 所示。

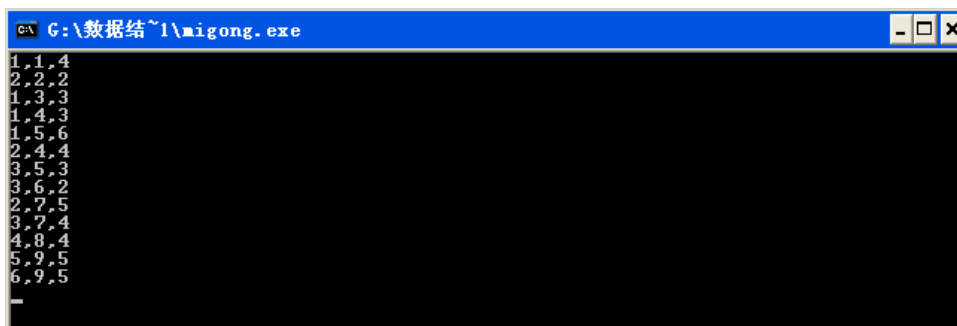


图 16-12 范例 16.10 结果图



16.10 关键路径

【范例 16.11】现有 5 个城市，每个城市到另一个城市的距离都不同。编写一个程序，求出其关键路径。

分析：关键路径是指经过所有点的最短路径，因此本题求 5 个城市的关键路径即为求经过这 5 个城市的最短路径问题。

范例 16.11 代码实现

```

01     #define max 100
02     #define m 5
03     int cost[m][m]={300, 10,300, 30, 100,          /*初始化各个城市之间的距离*/
04                     300,300, 50,300,300,
05                     300,300,300,300, 10,
06                     300,300, 20,300,60,

```



```
07         300,300,300,300,300};
08 int dist[5],path[5] ,s[5];          /*定义所需数组*/
09 void shortpath(int n,int v0)        /*最短路径求解函数*/
10 {
11     int num,w,min,u,i,j;
12     for(i=0;i<n;++i)                /*输出各个城市之间的距离*/
13     {
14         for(j=0;j<n;++j)
15             printf("%7d",cost[i][j]);
16         printf("\n");
17     }
18     for(w=1;w<=n-1;++w)              /*初始化数组*/
19     {
20         dist[w]=cost[v0][w];
21         if(cost[v0][w]<32767)
22             path[w]=v0;
23     }
24     for(w=0;w<=n-1;++w)              /*初始化数组 s*/
25         s[w]=0;
26     s[v0]=1;
27     num=1;
28     while(num<=n-1)                  /*while 循环，执行条件为 num 小于 n-1*/
29     {
30         min=32767;
31         u=v0;
32         for(w=1;w<=n-1;++w)          /*比较各个城市间的距离*/
33             if(s[w]==0&&dist[w]<min)
34             {
35                 u=w;
36                 min=dist[w];
37             }
38         s[u]=1;num++;
39         printf("min=%d,u=%d\n",min,u); /*输出到城市的最短距离*/
40         for(w=1;w<=n-1;++w)
41         {
42             if(s[w]==0&&dist[u]+cost[u][w]<dist[w]) /*将城市间最短距离赋给 dist 数组*/
43             {
44                 dist[w]=dist[u]+cost[u][w];
45                 path[w]=u;                /*将城市编号赋给 path 数组*/
46             }
47         }
48         for(w=1;w<=n-1;w++)           /*输出 dist 数组的值*/
49             printf("%7d",dist[w]);
50         printf("\n") ;
51     }
52 }
53 void printpath(int n,int v0)          /*输出函数*/
```



```

54  {
55      int i,k;
56      for(i=1;i<=n-1;++i)
57      {
58          if(s[i]==1)
59          {
60              k=i;
61              while(k!=v0)                /*输出最短路径*/
62              {
63                  printf("%5d",k);
64                  k=path[k];
65              }
66              printf("%5d",k);
67              printf("    length:  %5d",dist[i]);    /*输出最短距离*/
68          }
69          else                            /*若找不到路径输出以下信息*/
70          {
71              printf(i,v0);
72              printf("∞");
73          }
74          printf("\n") ;
75      }
76  }
77  void main()
78  {
79      shortpath( 5,0);                    /*调用自定义函数 shortpath()*/
80      printpath( 5,0);                    /*调用自定义函数 printpath()*/
81  }

```

【代码分析】本例求解关键路径问题，详细代码分析如下：

- 第 3~7 行，利用 cost 数组初始化各个城市之间的距离。
- 第 9~52 行，自定义函数 shortpath()，用来求解最短路径问题。
- 第 53~76 行，自定义函数 printpath()，用来输出求解出的最短路径至屏幕。

【运行结果】该程序的执行结果如图 16-13 所示。

```

C:\数据结~1\7-4.exe
300  10  300  30  100
300  300  50  300  300
300  300  300  300  10
300  300  20  300  60
300  300  300  300  300
min=10,u=1
10  60  30  100
min=30,u=3
10  50  30  90
min=50,u=2
10  50  30  60
min=60,u=4
10  50  30  60
1, 0 length: 10
2, 3, 0 length: 50
3, 0 length: 30
4, 2, 3, 0 length: 60

```

范例 16-13 范例 16.11 结果图



16.11 小结

本章讲解了 C 语言中一些常见的经典问题，读者应认真研究和學習，将有助于提高编程能力和思维能力。有时一个问题可以有多种不同的解决方法，不要仅仅局限于一种方法，可以尝试多种方法来解决问题。



16.12 习题

一、选择题

1. 以下程序运行结果为 ()。

```
void fun(char *c)
{
    if(*c>='a'&&*c<='z')
        *c=*c-32;
    else if(*c>='A'&&*c<='Z')
        *c=*c+32;
    printf("%c",*c);
}
void main()
{
    char c='f';
    fun(&c);
}
```

A. f

B. F

C. f 或 F

D. 以上结果都不正确

【提示】上述程序将字符 c 传递给函数 f(), 该函数将小写字母转化为大写字母，将大写字母转化为小写字母。

2. 设有以下程序，则下列表达式中错误的是 ()。

```
#define m 6;
void main()
{
    int x=1;
    float y=2.5;
    char z='f'
}
```

A. x++

B. y++

C. z++

D. m++

【提示】宏定义不能改变其值，只能取消后重新进行定义。



3. 以下程序的运行结果为 ()。

```
void main()
{
    int x=101,y=011;
    printf("%2d,%2d\n",x,y);
}
```

A. 101,11

B. 101,9

C. 01,9

D. 01,11

【提示】上述程序中 101 为十进制数，011 为八进制数，需转化为十进制数。

4. 现有以下程序，在运行时输入 ba，则其输出结果为 ()。

```
void main()
{
    char c;
    int i;
    for(i=0;i<2;i++)
    {
        scanf("%c",&c);
        switch(c)
        {
            case 'a':printf("first\n");
            case 'b':printf("second\n");
        }
    }
}
```

A. second

B. second

first

first

second

C. first

D. second

second

first

second

first

【提示】上述程序本题通过 switch 结构进行简单的输出，其中要注意有无 break 语句。

5. 现有以下程序，则运行结果为 ()。

```
void main()
{
    int a=0,b=2,c=3;
    while(c--&&b++)
        a++;
    printf("%d,%d,%d",a,b,c);
}
```

A. 3,5,0

B. 3,5,-1

C. 3,4,0

D. 3,5,0

【提示】在 while 循环中若 c--&&b++ 值为 1，则会一直执行直到其值为 0 退出。



6. 以下程序的运行结果为 ()。

```
void main()
{
    int i,s=0;
    for(i=1;i<=5;i++)
    {
        do
        {
            s++;
            if(i%4==0) continue;
        }while(!i);
    }
    printf("%d",s);
}
```

A. 4

B. 5

C. 6

D. 9

【提示】本题中!i 值为真, 则 do-while 循环会一直执行, 始终执行 s++操作。

7. 下列程序的运行结果为 ()。

```
#include <stdlib.h>
struct node
{
    int n;
    struct node *next;
};
void main()
{
    struct node *x,*y,*z;
    int s=0;
    x=(struct node *)malloc(sizeof(struct node));
    y=(struct node *)malloc(sizeof(struct node));
    z=(struct node *)malloc(sizeof(struct node));
    x->n=5;
    y->n=6;
    z->n=7;
    x->next=y;
    y->next=z;
    z->next=NULL;
    s=x->next->n+y->next->n+z->n;
    printf("%d",s);
}
```

A. 18

B. 19

C. 20

D. 21

【提示】本题通过结点进行简单的算术运算后, 输出结果至屏幕。

8. 以下程序运行结果为 ()。



```
void main()
{
    int y=20,i=0,j,s[8];
    do
    {
        s[i]=y%2;
        i++;
        y=y/2;
    }while(y>1);
    for(j=i-1;j>0;j--)
        printf("%d ",s[j]);
}
```

A. 0 1 0

B. 0 0 1 0

C. 1 0 0

D. 1 1 0

【提示】本题将 $y\%2$ 的余数存放在 s 数组中直到 y 的值小于或等于 1 为止，最后倒着输出 s 数组中的内容。

9. 以下程序运行结果为 ()。

```
int fun(int x,int y)
{
    static int s=0,z=3;
    z=z+s-1;
    s=s+x+y+z;
    return s;
}
void main()
{
    int x=5,y=6,z;
    z=fun(x,y);
    printf("%d ",z);
    z=fun(x,y);
    printf("%d ",z);
}
```

A. 13 30

B. 13 38

C. 12 30

D. 14 38

【提示】本题通过静态变量及动态变量进行算术运算，输出结果至屏幕。

10. 以下程序运行时输入 135 空格 246 空格 567 回车，则其输出结果为 ()。

```
void main()
{
    char s[10];
    int x;
    char c;
    scanf("%c",&c);
    scanf("%d",&x);
}
```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
scanf("%s",s);  
printf("%c,%d,%s\n",c,x,s);  
}
```

A. 1,246,567

B. 135,246,567

C. 1,35,246

D. 13,246,567

【提示】从键盘输入数据时，不同的类型变量只能获取该类型的数据。

11. 以下程序运行结果为（ ）。

```
struct node  
{  
    int x;  
    char y;  
};  
void main()  
{  
    struct node n[3];  
    printf("%d",sizeof(n));  
}
```

A. 8

B. 9

C. 10

D. 11

【提示】本题定义了结构体数组 s，然后调用 sizeof 运算符求出其存储空间并输出。

12. 以下程序运行结果为（ ）。

```
void fun1(char x,char y)  
{  
    char t;  
    t=x;  
    x=y;  
    y=t;  
}  
void fun2(char *x,char y)  
{  
    char t;  
    t=*x;  
    *x=y;  
    y=t;  
}  
void fun3(char *x,char *y)  
{  
    char t;  
    t=*x;  
    *x=*y;  
    *y=t;  
}  
void main()
```



```
{
    char a='f',b='D';
    fun1(a,b);
    printf("%c %c ",a,b);
    fun2(&a,b);
    printf("%c %c ",a,b);
    fun3(&a,&b);
    printf("%c %c ",a,b);
}
```

A. fDDDDDD

B. ffDDDD

C. DDDDDDD

D. fffDDD

【提示】本题中包含 f1()、f2()、f3() 这三个函数，不同的函数具有不同的功能。

13. 以下程序运行结果为 ()。

```
void fun(int x,int y)
{
    int t;
    while(y)
    {
        t=y%x;
        y=y/x;
        printf("%d",t);
    }
}

void main()
{
    int a=5,b=9;
    fun(a,b);
}
```

A. 39

B. 40

C. 41

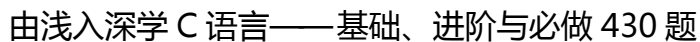
D. 42

【提示】本题通过 fun() 函数求 y%x 的余数，将结果输出至屏幕。

14. 以下程序运行结果为 ()。

```
void fun(char c)
{
    if(c>='a'&&c<='z')
        c=c-32;
    if(c>='A'&&c<='Z')
        c=c+32;
    printf("%c",c);
}

void main()
{
    char c='f';
```



D. 以上结果都不正确



```
printf("%s",q);  
}
```

【提示】本题通过 f() 和 strlen() 函数计算字符串长度与其长度除以 3 值的和。

4. 下列程序的运行结果为_____。

```
int f(int x[],int n)  
{  
    if(n>1)  
        return x[0]+f(&x[n-1],--n);  
    else  
        return x[0];  
}  
void main()  
{  
    int x[]={1,2,3,4,5,6},y;  
    y=f(x,6);  
    printf("%d",y);  
}
```

【提示】本题通过递归调用 f() 函数，最后返回结果输出至屏幕。

5. 以下程序的运行结果为_____。

```
#include <string.h>  
typedef struct s  
{  
    char n[10];  
    char number[10];  
    float score;  
}stu;  
void main()  
{  
    stu student1={"zhangli",232131,89.5},student2={"huanchao",214124,90.5};  
    struct s x;  
    struct s *p=&x;  
    x=student1;  
    if(strcmp(x.n,student2.n)<0)  
        x=student2;  
    printf("%s,%s,%f\n",x.n,x.number,p->score);  
}
```

【提示】上述程序定义了结构体变量 student1 和 student2，然后比较其大小再进行输出。

6. 以下程序运行结果为_____。

```
void main()  
{  
    char s[]="abcasdef#";  
    char *p;
```




```
p=s;
while(*p!='#')
{
    printf("%c",*p);
    p++;
}
}
```

【提示】本题输出字符数组 s 中的内容直到遇到#号为止。

7. 以下程序运行结果为_____。

```
void main()
{
    int x=1,y=2;
    while(x-->0||y++<6)
        y++;
    printf("%d,%d",x,y);
}
```

【提示】本题通过 while 循环使 y 值始终加 1，直至 x-->0||y++<6 真值为 0 为止。

8. 以下程序运行结果为_____。

```
void main()
{
    int x[5]={1,4,8,2,9};
    int *p;
    p=x+3;
    printf("%d",*p);
}
```

【提示】上述程序中定义整型数组 x 并进行初始化，然后定义指针 p 指向数组的第 4 个元素输出至屏幕。

9. 以下程序运行结果为_____。

```
void main()
{
    struct node
    {
        int data;
        struct node *next;
    }s[4];
    int i;
    for(i=0;i<4;i++)
        s[i].data=i;
    printf("%d",s[i].data);
}
```

【提示】上述程序定义结构体，然后进行相应的输出。



10. 以下程序运行结果为_____。

```
void swap1(int x,int y)
{
    int t;
    t=x;
    x=y;
    y=t;
}
void swap2(int x,int y)
{
    int t;
    t=x;
    x=y;
    y=t;
}
void main()
{
    int a=5,b=6;
    swap1(a,b);
    printf("%d,%d ",a,b);
    swap2(a,b);
    printf("%d,%d ",a,b);
}
```

【提示】上述程序中 swap1()函数和 swap2()函数都是传值调用，因此其值变化都不会影响主函数中 a 和 b 的值。

11. 以下程序运行结果为_____。

```
void main()
{
    char s[]="ajbncesf";
    char *p;
    p=s+4;
    printf("%d",*p);
}
```

【提示】本题定义字符数组 s 并进行初始化，然后定义指针 p 指向数组的第 5 个字符输出至屏幕。

12. 以下程序运行结果为_____。

```
void main()
{
    int i,s=0;
    int a[]={1,3,5,7,8};
    for(i=0;i<5;i++)
        s=s+i*a[i];
}
```



```
printf("%d",s);  
}
```

【提示】上述程序通过 for 循环计算 $i*a[i]$ 的累加和，其中 i 从 0 变化至 4。

13. 以下程序运行结果为_____。

```
struct node  
{  
    int x;  
    float score;  
};  
void fun1(struct node x)  
{  
    struct node y={41,78.6};  
    x=y;  
}  
void fun2(struct node *x)  
{  
    struct node y={50,84.6};  
    *x=y;  
}  
void main()  
{  
    struct node a={40,70.8};  
    fun1(a);  
    printf("%d,%d\n",a.x,a.score);  
    fun2(&a);  
    printf("%d,%d\n",a.x,a.score);  
}
```

【提示】上述程序中 fun1() 函数为传值调用，fun2() 函数为传址调用，因此 fun1() 函数不会影响结点 a 的值，fun2() 函数会影响结点 a 的值。

14. 以下程序运行结果为_____。

```
void main()  
{  
    struct node  
    {  
        int a[3];  
        double b;  
    }s[2];  
    printf("%d",sizeof(s));  
}
```

【提示】本题定义一个结点数组 s，然后通过 sizeof 运算符计算其存储空间大小后输出。

15. 以下程序运行结果为_____。

```
void max(int a[])
```



```
{
    int i,j,t;
    for(i=0;i<3;i++)
        for(j=i+1;j<4;j++)
            if(a[i]>a[j])
            {
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
}

void main()
{
    int i,a[5]={1,8,4,7,2};
    max(a);
    for(i=0;i<5;i++)
        printf("%d ",a[i]);
}
```

【提示】上述程序 max() 函数通过冒泡排序的思想对数组 a 进行部分排序，然后调用 for 循环输出数组中的元素。

三、编程题

1. 从键盘输入三个学生的信息，从中找出分数最低的学生并输出该学生的信息至屏幕。

【提示】定义一个变量，比较这三名学生的分数，保存其中分数最低的学生下标，然后通过该下标输出该名学生的信息。

【核心代码】

```
struct
{
    char name[10];
    char num[10];
    float score;
}s[3];
for(i=0;i<2;i++)
    if(s[i].score<s[i+1].score)
        j=i;
```

2. 从键盘输入 10 个数，利用冒泡排序将其从小到大排序。

【提示】冒泡排序的思想为比较相邻的数，从而将较小（或较大）的数往前移动，第一趟比较将最小（或最大）的数移动至序列的最前方，如此进行过 n-1 趟即可得到排好序的序列。

【核心代码】

```
for(i=0;i<10;i++)
    scanf("%d",&x[i]);
for(i=0;i<9;i++)
```



```
for(j=i+1;j<10;j++)
if(x[i]>x[j])
{
    t=x[i];x[i]=x[j];x[j]=t;
}
```

3. 从键盘输入 5 个数，建立一个链表，然后再从键盘输入一个数，按数值的大小插入至链表相应的位置。

【提示】创建一个链表，可以用动态分配内存的方法实现。然后通过后继指针连接起来，形成一个链表。插入结点前，设置一个指针用于搜索和比较数值的大小，同时定义另外一个指针用于保存指针的前一个结点位置。

【核心代码】

```
void create(link *l)
{
    (*l)=(link)malloc(sizeof(node));
    (*l)->next=NULL;
    q=*l;
    for(i=0;i<5;i++)
    {
        p=(link)malloc(sizeof(node));
        scanf("%d",&p->data);
        p->next=NULL;
        q->next=p;
        q=p;
    }
}

void insert()
{
    scanf("%d",&n);
    while(p->data<n)
    {
        q=p;
        p=p->next;
    }
    r=q->next;
    q->next=n;
    n->next=r;
}
```

4. 编写程序，删除一个文件中的注释部分。

【提示】文件注释是以“/*”和“*/”为标志的，对文件以写的方式打开。然后判断文件是否有“/*”开头，“*/”为结尾的内容，记录其位置然后删除相应的内容。

【核心代码】

```
fp=fopen(.....,"w+");
```



```
ch=getc(fp);
while(!feof(fp))
{
    chl=fgetc(fp);
    if(ch=='/'&&chl=='*')
        p1=ftell(fp)-2;
    if(ch=='*'&&chl=='/')
    {
        p2=ftell(fp)-1;
        fseek(fp,p1,1);
    }
    ch=chl;
}
```

5. 从键盘输入 10 个数，利用选择排序法将其从小到大排序。

【提示】选择排序法是通过变量先保存其下标，然后将该元素与其他元素比较，若大于其他元素，则将下标赋值给该变量，最后比较该变量是否初始值相同。若值改变则交换两个元素，否则不变。

【核心代码】

```
for(i=0;i<9;i++)
{
    k=i;
    for(j=i+1;j<10;j++)
        if(x[i]>x[j])
            k=j;
    if(k!=i)
    {
        temp=x[k];
        x[k]=x[i];
        x[i]=temp;
    }
}
```

6. 编写一个学生管理系统，要求能实现以下功能：

- (1) 实现学生信息的输入。
- (2) 能根据学生的姓名和学号查找出学生的信息。
- (3) 可以根据学生的成绩总分进行排序再输出。

【提示】学生的信息由于包含多种类型的数据，因此应定义为结构体类型。然后将输入、查询、排序功能分别编写成不同的函数，再在主函数中调用这些函数。

【核心代码】

```
struct student
{
    char name[9];
    char number[10];
```



```
float math;
float english;
};
do
{
    printf("\n 输入第%d 个学生的信息:\n 输入学生姓名: ",count+1);
    scanf("%s",arr[count].name);
    printf("\n 输入学生学号: ");
    scanf("%s",arr[count].number);
    printf("\n 输入数学成绩: ");
    scanf("%f",&arr[count].math);
    printf("\n 输入英语成绩: ");
    scanf("%f",&arr[count].english);
    count++;
    printf("\n 是否继续输入(y/n)\n");
    fflush(stdin);
    ch=getchar();
}while(ch=='y' || ch=='Y');
void sort(struct student *arr)
{
    if(count==0)
    printf("无学生成绩, 请先录入学生信息\n");
    else
    {
        for(i=0;i<count-1;i++)
        {
            k=i;
            for(j=i+1;j<count;j++)
            if((arr[k].math+arr[k].english)<(arr[j].math+arr[j].english))
            k=j;
            if(k!=i)
            {
                t=arr[k];
                arr[k]=arr[i];
                arr[i]=t;
            }
        }
    }
}
void find(struct student *arr)
{
    scanf("%s",name);
    scanf("%s",number);
    for(j=0;j<count;j++)
    {
        if((strcmp(name,arr[j].name)==0)&&(strcmp(number,arr[j].number)==0))
```



```

{
    printf("\n\t姓名\t学号\t数学\t英语\t总分\n");
    printf("\t%-6s\t%-4s\t%-6.1f\t%-6.1f\t%-6.1f\n",
        arr[j].name, arr[j].number, arr[j].math, arr[j].english, arr[j].math+arr[j].english);
    break;
}
}
}

```

7. 有两个磁盘文件 A 和 B, 各存放一行字母, 要求把这两个文件中的信息合并 (按字母顺序排列), 输出到一个新文件 C 中。

【提示】将两个文件的信息合并, 可以设置两个文件指针, 分别指向这两个文件的开头, 然后逐个字符进行比较, 将其中较小的字符写至文件中。

【核心代码】

```

for(i=0;(ch=fgetc(fp))!=EOF;i++)
{
    c[i]=ch;
    putchar(c[i]);
}
ni=i;
for(i=0;(ch=fgetc(fp))!=EOF;i++)
{
    c[i]=ch;
    putchar(c[i]);
}
n=i;
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(c[i]>c[j])
{t=c[i];c[i]=c[j];c[j]=t;}
printf("\n C file is:\n");
fp=fopen("C","w");
for(i=0;i<n;i++)
{
    putc(c[i],fp);
    putchar(c[i]);
}

```

8. 从键盘输入一个字符串, 将小写字母全部转换成大写字母, 然后输出到一个磁盘文件“test”中保存, 输入的字符串以!结束。

【提示】将从键盘输入的字符串保存至字符数组中, 然后对每个字符进行判断, 若为小写字母则减去 32 转化为大写字母, 再将字符写至 test 文件中, 直到遇到!为止。

【核心代码】

```

if((fp=fopen("test","w"))==NULL)

```




```
{
printf("cannot open the file\n");
exit(0);
}
printf("please input a string:\n");
gets(str);
while(str[i]!='!')
{
    if(str[i]>='a'&&str[i]<='z')
    str[i]=str[i]-32;
    fputc(str[i],fp);
    i++;
}
fp=fopen("test","r");
fgets(str,strlen(str)+1,fp);
printf("%s\n",str);
```

9. 编写一个猜数游戏，判断一个人的反应快慢。

【提示】随机产生一个数可以通过 `rand()` 函数来实现，然后调用 `time.h` 头文件中的函数计时直到用户猜出正确的数为止，输出所花费的时间至屏幕。

【核心代码】

```
srand(time(NULL));
printf("要开始猜数吗('y' or 'n') \n");
loop:
while((c=getchar())=='y')
{
    i=rand()%100;
    printf("\n 输入猜的数字:\n");
    start=clock();
    a=time(NULL);
    scanf("%d",&guess);
    while(guess!=i)
    {
        if(guess>i)
        {
            printf("please input a little smaller.\n");
            scanf("%d",&guess);}
        else
        {
            printf("please input a little bigger.\n");
            scanf("%d",&guess);}
    }
    printf("\1: It took you %6.3f seconds\n",var=(double)(end-start)/18.2);
    printf("\1: it took you %6.3f seconds\n\n",difftime(b,a));
    if(var<15)
```



```
printf("\1\1 You are very clever! \1\1\n\n");
else if(var<25)
printf("\1\1 you are normal! \1\1\n\n");
else
printf("\1\1 you are stupid! \1\1\n\n");
printf("\1\1 Congradulations \1\1\n\n");
printf("The number you guess is %d",i);
}
printf("\n 重新再来?(\\"y\".or.\"n\")\n");
if((c=getch())=='y')
goto loop;
```

10. 综合 C 语言所学知识, 编写一个家庭财务管理小程序。

【提示】家庭财务管理系统应包含家庭收入、开支等方面的信息, 同时可以利用图形界面的知识在屏幕上显示出来。

【核心代码】

```
getdate(&d);
fprintf(ctime,"%4d.%02d.%02d",d.da_year,d.da_mon,d.da_day);
for(;;)
{
gotoxy(3,24);printf(" Tab __browse cost list Esc __quit");
gotoxy(13,10);printf(" ");
gotoxy(13,13);printf(" ");
gotoxy(13,7);printf("%s",ctime);
if(ch[0]==9)
{
fp=fopen("home.dat","r+");
gotoxy(3,24);printf(" ");
gotoxy(6,4);printf(" list records ");
gotoxy(1,5);printf("|-----|");
gotoxy(41,4);printf(" ");
gotoxy(41,5);printf(" |");
if ((i%36)<17)
{
gotoxy(4,6+i);
printf(" ");
gotoxy(4,6+i);}
else if((i%36)>16)
{
gotoxy(41,4+i-17);
printf(" ");
gotoxy(42,4+i-17);}
i++;
sum=sum+chm;
printf("%10s %-14s %6.1f\n",ctime,chshop,chm);}
```



```
gotoxy(1,23);printf(" |-----|");
gotoxy(1,24);printf(" |");
gotoxy(1,25);printf(" |-----|");
gotoxy(10,24);printf("total is %8.1f$",sum);
gotoxy(49,24);printf("press any key to.....");
}
else
{
while(ch[0]!='\r')
{
    if(j<10)
    {
        strncat(chtime,ch,1);
        j++;
    }
    if(ch[0]==8)
    {
        len=strlen(chtime)-1;
        if(j>15)
        {
            len=len+1; j=11;
        }
        strcpy(ch1,"");
        j=j-2;
        strncat(ch1,ctime,len);
        strcpy(ctime,"");
        strncat(ctime,ch1,len-1);
        gotoxy(13,7);printf(" ");}
        gotoxy(13,7);
        printf("%s",ctime);ch[0]=getch();
        if(ch[0]==9)
            goto mm;
        if(ch[0]==27)
            exit(1);
    }
    gotoxy(3,24);printf(" ");
    gotoxy(13,10);
    j=0;
    ch[0]=getch();
    while(ch[0]!='\r')
    {
        if (j<14)
        {
            strncat(chshop,ch,1);
            j++;
        }
    }
}
```



```
if(ch[0]==8)
{
    len=strlen(chshop)-1;
    strcpy(ch1,"");
    j=j-2;
    strncat(ch1,chshop,len);
    strcpy(chshop,"");
    strncat(chshop,ch1,len-1);
    gotoxy(13,10);printf(" ");}
gotoxy(13,10);printf("%s",chshop);ch[0]=getch();
}
gotoxy(13,13);
j=0;
ch[0]=getch();
while(ch[0]!='\r')
{
    if (j<6)
    {
        strncat(chmoney,ch,1);
        j++;
    }
    if(ch[0]==8)
    {
        len=strlen(chmoney)-1;
        strcpy(ch1,"");
        j=j-2;
        strncat(ch1,chmoney,len);
        strcpy(chmoney,"");
        strncat(chmoney,ch1,len-1);
        gotoxy(13,13);printf(" ");}
        gotoxy(13,13);printf("%s",chmoney);ch[0]=getch();
    }
    if((strlen(chshop)==0)|| (strlen(chmoney)==0))
        continue;
    if((fp=fopen("home.dat","a+"))!=NULL);
    fprintf(fp,"%10s%14s%6s",ctime,chshop,chmoney);
    fputc('\n',fp);
    i++;
    gotoxy(41,5+i);
    printf("%10s %-14s %-6s",ctime,chshop,chmoney);
}
}
}
```

11. 编写一个程序，建立一个链表，然后将其逆置后输出。

【提示】从键盘输入表中元素值，编写一个函数利用尾插法建单链表，并返回该单链表头



指针 L，然后通过指针逆序输出链表中的所有元素。

【核心代码】

```
void CreateFromTail(LinkList *L)
{
    *L=(Node * )malloc(sizeof(Node));
    (*L)->next=NULL;
    r=*L;
    while(flag)
    {
        c=getchar();
        if(c!='$')
        {
            s=(Node*)malloc(sizeof(Node));
            s->data=c;
            r->next=s;
            r=s;
        }
        else
        {
            flag=0;
            r->next=NULL;
        }
    }
}

void ReverseList(LinkList *L)
{
    p=(*L)->next;
    (*L)->next=NULL;
    while(p!=NULL)
    {
        q=p->next;
        p->next=(*L)->next;
        (*L)->next=p;
        p=q;
    }
}

CreateFromTail(&l);
ReverseList(&l);
printf("逆置后的单链表为:\n");
p = l->next;
while(p!=NULL)
{
    printf("%c",p->data);
    p=p->next;
}
```



12. 编写一个程序，实现链表的插入和删除功能。

【提示】首先通过结点创建一个链表，将链表的插入和删除功能写成单独的函数，然后在主函数中调用对应的函数，其中要将链表的首地址作为参数传递给函数，函数才能对链表中的数据进行插入和删除操作。

【核心代码】

```
typedef struct lnode
{
    int data;
    struct lnode *next;
}lnode,*linklist;
void createlist (linklist *l)
{
    (*l)=(linklist)malloc(sizeof(lnode));
    (*l)->next=NULL;
    printf("Please enter the node's amount:");
    scanf("%d",&n);
    for(i=n;i>0;i--)
    {
        linklist p;
        p=(linklist)malloc(sizeof(lnode));
        printf("Please enter a number for the node:");
        scanf("%d",&p->data);
        p->next=(*l)->next;
        (*l)->next=p;
    }
}
int linkinsert(linklist *l,int i,int e)
{
    p=*l;
    while(p&&j<i-1) {p=p->next;++j;}
    if(!p||j>i-1) return 0;
    s=(linklist)malloc(sizeof(lnode));
    s->data=e;
    s->next=p->next;
    p->next=s;
    return 1;
}
void linkdelete(linklist *l,int i,int *e)
{
    while(p->next&&j<i-1)
        p=p->next,++j;
    if(!(p->next)||j>i-1) ;
    else
    {
        q=(linklist)malloc(sizeof(lnode));
```



```
    q=p->next;
    p->next=q->next;
    *e=q->data;
    free(q);
}
}
void print(linklist l)
{
    l=l->next;
    while(l)
    {
        printf("%d ",l->data);
        l=l->next;
    }
    printf("\n");
}
```

13. n 个人座一圈，报 m 号出圈，最后一个报 m 号的人胜利，编写一个程序求解该问题。

【提示】 n 个人坐一圈报数，可以通过链表来模拟。定义 n 个结点并初始化，设置一个计数器进行报数，若计数器的值等于 m 则那个人退出圈内，计数器置 0，然后重新报数，直到剩下最后一个人为止停止报数。

【核心代码】

```
typedef struct QNode
{
    int data;
    struct QNode *next;
}QNode,*linklist;
void create(linklist *L,int n)
{
    *L=(QNode*) malloc(sizeof(QNode));
    (*L)->next=NULL;
    q=*L;
    for(i=1;i<=n;++i)
    {
        p=(QNode*) malloc(sizeof(QNode));
        p->data=i;
        q->next=p;q=p;
    }
    q->next=(*L)->next;
    *L=(*L)->next;
}
void print(linklist L)
{
    while(p->next!=L)
    {
```



```
printf("%5d",p->data);
p=p->next;
}
printf("%5d",p->data);
printf("\n");
}
void jone(linklist L,int m)
{
p=L;
while(p->next!=p)
{
j=1;
while(p->next!=p&&j<m-1)
{
p=p->next;
++j;
}
q=p->next;
printf("%5d",q->data);
p->next=q->next;
p=p->next;
free(q);
}
printf("\n 最后一个:%5d",p->data);
}
```

14. 编写一个程序,实现双向链表的建立、插入和删除功能。

【提示】双向链表与单向链表不同,其具有两个指针,分别指向结点的前驱和后继。因此双向链表数据发生改变时,两个指针的内容都必须进行改变。

【核心代码】

```
typedef struct dubnode
{
int data;
struct dubnode *prior,*next;
}node,*dublink;
void crea(dublink *L,int n)
{
for(i=1;i<=n;++i)
{
p=(dublink)malloc(sizeof(node));
p->data=i;
p->next=NULL;
p->prior=q;
q->next=p;
q=p;
}
```




```
    }
}
void print(dublink L)
{
while(p)
{
    printf("%5d",p->data);
    p=p->next;
}
}
void insert(dublink *L,int i,int e)
{
while(p&& j<i)
{
    p=p->next;
    ++j;
}
if(!p||j>i)
printf("error");
else
{
    s= (dublink)malloc(sizeof(node));
    s->data=e;
    s->prior=p->prior;
    p->prior->next=s;
    s->next=p;
    p->prior=s;
}
}
void de(dublink *L,int i)
{
while(p&& j<i)
{
    p=p->next;
    ++j;
}
if(!p||j>i)
printf("error");
else
{
    s=p;
    p->prior->next=p->next;
    p->next->prior=p->prior;
    free(s);
}
}
```



15. 现有 n 个孩子围成一圈分糖果, 老师先随机地发给每个孩子若干颗糖果, 然后按以下规则调整: 老师每次一吹口哨, 每个孩子同时将自己手中的糖果分一半给坐在他右边的小朋友。如共有 8 个孩子, 则第 1 个将原来的一半分给第 2 个, 第 2 个将原有的一半分给第 3 个……第 8 个将原来的一半分给第 1 个, 这样的平分动作同时进行, 要求平分之前每个孩子的糖果数必须是偶数个, 若平时孩子糖果数为奇数, 则可以向老师多要一个糖果。小孩人数和每个小孩的糖果数由键盘输入。问老师吹了几次口哨, 能使每个孩子手中的糖果一样多, 每个孩子手中的糖果数为多少。

【提示】孩子个数 n 由键盘输入, 然后将每个孩子的糖果数保存至数组中, 每吹一次口哨将孩子糖果数的一半给下一个孩子, 通过循环重复执行此操作直到每一个孩子的糖果相等为止, 退出循环输出每个孩子的糖果数及吹口哨的次数。

【核心代码】

```
scanf("%d",&N);
for(i=1;i<=N;i++)
{
    scanf("%d",&a[i]);
    if(a[i]%2!=0)
    {
        printf("输入偶数");
        break;
    }
}
while(1)
{
    t=0;
    for(j=2;j<=N;j++)
    {
        if(a[1]==a[j])
            t++;
    }
    if(t==N-1)
        break;
    flag++;
    m=a[N]/2;
    for(n=N;n>=2;n--)
        a[n]=a[n-1]/2+a[n]/2;
    a[1]=m+a[1]/2;
    for(m=1;m<=N;m++)
        if(a[m]%2==1)
            a[m]=a[m]+1;
}
```

第 17 章 C 语言开发项目： 航空订票管理系统

经过前面章节的学习，读者应基本掌握了 C 语言的知识及其应用。本章将结合 C 语言的知识讲解如何利用 C 语言来开发航空管理系统。

本章主要涉及的知识点有：

- 航空订票管理系统需求分析；
- 航空订票管理系统可行性分析；
- 航空订票管理设计。



17.1 航空订票管理系统简介

随着社会的不断发展，乘坐飞机的人越来越多，有关航空方面的管理越来越繁杂。面对庞大的顾客信息流量，需要有专门的航空管理系统来提高航空管理工作的效率，规范信息管理及实现航班的快速查询、修改、增加、删除等功能，从而减少管理方面的工作量。

航空订票管理系统主要用于航班信息的管理，总体任务是实现顾客订票的系统化、规范化和自动化，其主要任务是用计算机对每个航班的信息进行日常管理，如查询、修改、增加、删除等操作，另外还包括顾客订票、退票等。

本系统主要包括航班信息查询、订票和退票三部分。其功能主要如下：

- (1) 有关航班信息的输入，包括输入航班编号、起始点、终点和时间等。
- (2) 航班信息的查询，包括查询航班编号、起始地点、目的地和时间等。
- (3) 航班信息的修改。
- (4) 航班信息的删除，即删除整个航班信息。
- (5) 客户订票功能。
- (6) 客户退票功能。
- (7) 添加航班信息的功能。

经过分析，在本章中使用微软公司的 Visual C++ 开发工具，利用其提供的各种面向对象的开发工具及 C 语言知识进行系统的开发设计。



17.2 航空订票系统需求分析

航空订票管理系统是每个机场不可缺少的部分。一个功能齐全、简单易用的航空管理系统能有效地减轻机场相关工作人员的工作负担，其功能对于顾客能否快速订票至关重要。所以航



空管理系统应能为用户提供充足的航班信息和快捷的查询手段。

以前人们一直使用传统的人工的方法来管理文件档案、统计和查询航班数据，这种管理方式存在着许多缺点。例如工作效率低、保密性低等，另外时间越长，文件和数据堆积的越来越多，这对于查找、更新和维护航班的信息都会带来很大的困难。随着科学技术的不断发展，计算机科学逐渐成熟，计算机强大的功能已被人们深刻认知，它已进入社会的各个领域并发挥着至关重要的作用。

作为计算机应用的一部分，使用计算机对航班的各类信息进行管理，具有人工管理无法比拟的优点。例如，查询迅速、工作效率高、可靠性好、存储量大、保密性高、寿命长和成本低等。这些优点能够在很大的程度上提高机场工作人员的效率，具有很好的前景性。因此开发一套具有完整的查询航班、订票、退票等功能的航空管理系统势在必行。

机票管理系统是为机场工作人员和客户提供的实现订票退票等功能的系统软件，它应具有易维护、易扩充、交互性好等优点。它能为用户提供完整、精确的航班信息，能快速实现客户订票和退票，方便机场工作人员对航班信息的管理。



17.3 航空订票系统可行性分析

对于机场而言，有大量的航班信息需要修改和记录，包括航班编号、起始点、目的地、所需时间、航班票数等。这些信息可能会随时发生改变，需要及时进行修改，若采用人工的方式进行修改，则一天的工作量会很大。

航空订票管理系统是为顾客和机场工作人员开发的，本系统所采用的语言是 C 语言，在 Visual C++ 环境下调试完成。该系统总体由 4 部分组成，包括航班信息查询、订票功能、退票功能、添加和删除航班。通过本系统，可以对航班信息进行统一的管理和修改，可以快捷地提供给用户最新的航班信息。

采用本系统，能使所有航班信息的管理工作简化，提高机场工作人员的工作效率。由于采用统一的结构存储，能够使客户及工作人员快速地查询所需的数据、资料及其他信息，使信息快速高效地运行。



17.4 航空订票系统总体设计

由于考虑到航班包含多种类型的数据，因此定义结构体保存航班信息，其定义形式如下：

```
struct node                                //定义结构体
{
    int bianhao;                            //定义航班编号
    char q[20];                             //定义航班起始点
    char o[20];                             //定义航班终点
    char t[20];                             //定义航班所需时间
    int n;                                  //定义航班票数
}s;
```



17.4.1 输入、输出航班信息

每个航班信息包含航班号、起始地、目的地、花费时间等信息，每个飞机管理员都应能对航班的信息进行管理操作。由于每个航班信息包含多种类型的数据，可以将其定义为结构体类型中，然后将数据分别保存至不同的结构体成员中。其实现代码如下：

```
printf("输入航班编号:\n");
scanf("%d",&s[i].bianhao);           //输入航班号
printf("输入航班起点:\n");
scanf("%s",s[i].q);                  //输入起始站
printf("输入航班目的地:\n");
scanf("%s",s[i].o);                  //输入终点站
printf("输入航班所需时间:\n");
scanf("%s",s[i].t);                  //输入时间
printf("输入航班的机票数:\n");
scanf("%d",&s[i].n);
```

17.4.2 订票功能

一个客户订票时，先查询相应的航班是否有余票。若有余票，则订票成功，该航班相应的票数减去用户订的票数；若没有余票，则提示该航班已满，不能订票的信息。其实现代码如下：

```
printf("输入要订的机票数: \n");
scanf("%d",&p);                       //输入所订机票数
if(p<0)
{
    printf("请输入有效的机票数! \n"); //判断机票数是否出错
    break;
}
if(s[i].n!=0&& s[i].n>=p)             //判断是否出错
{
    s[i].n=s[i].n-p;
    baocun();                          //调用保存函数
    printf("订票成功!\n\n");
    break;
}
else if(s[i].n<p)                     //判断机票数是否充足
{
    printf("机票不足: \n");
    break;
}
```

17.4.3 退票功能

一个客户退票时，应先查询是否存在相应的航班。若有相应的航班存在，则该航班的票数



加上用户退的票数并提示客户退票成功，若没有相应的航班，则提示该航班不存在的信息。其实现代码如下：

```
printf("输入要退的机票数: \n");
scanf("%d",&p);
if(p<0)
printf("请输入有效的机票数! \n");
if(p>0)
s[i].n=s[i].n+p;
```

//输入所退票数
//判断票数是否有效

17.4.4 查询航班信息

查询航班信息可以根据航班号、目的地、航班线路这三种方式来查找。例如根据终点站来查找航班，输入北京后，则可查询到所有到北京航班信息。其实现代码如下：

```
for(i=0;i<r;i++)
{
if(strcmp(s[i].q,n1)==0||strcmp(s[i].o,n2)==0)
{
printf("\n 查找到相应航班!\n");
printf("航班号 起点 目的地 花费时间 剩余机票数\n");
printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n); //输出信息
break;
}
if(s[i].num==t1)
{
printf("\n 查找到相应航班!\n");
printf("航班号 起始站 终点站 时间 机票数\n");
printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n); //输出信息
break;
}
}
```

//for 循环逐个比较查找
//按航班号判断输出条件

17.4.5 删除航班信息

添加一个航班信息，可以将信息保存至对应的结构体变量中。删除航班信息则可以通过航班编号、航班线路两种方式来删除，即将对应的结构体变量删除。其实现代码如下：

```
for(i=0;i<r;i++)
{
if(s[i].bianhao==t1||(strcmp(s[i].q,n1)==0&&strcmp(s[i].o,n2)==0))
{
s[i]=s[m-1];
m--;
}
}
```

//通过 for 循环逐个比较
//判断输入信息是否存在
//将结构体数组中的最后一个元素赋给 s[i]
//总航班数减去 1



其模块及功能划分如图 17-1 所示。

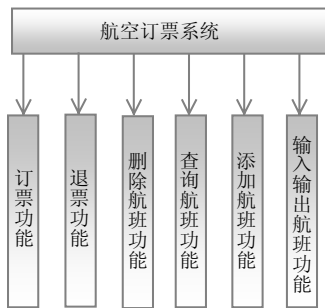


图 17-1 航空订票系统模块划分图



17.5 航空订票系统程序设计

【范例 17.1】综合利用所学的 C 语言知识，设计一个飞机票订票系统，要求具有以下功能：

- (1) 输入功能。
- (2) 输出功能。
- (3) 订票功能。
- (4) 退票功能。
- (5) 查询航班功能。
- (6) 添加航班功能。
- (7) 删除航班功能。

分析：由于每个航班包含时间、编号等多种数据，因此可以用结构体来保持每个航班的信息。然后将飞机订票系统的订票、退票、查询功能分别编写成不同的函数，编写在不同的文件中。最后在主函数中调用相应的函数从而实现不同的功能。

范例 17.1 代码实现

```
01  #include<stdio.h>                //标准输入、输出头文件
02  #include<string.h>               //包含字符串函数处理头文件
03  #include<stdlib.h>               //包含动态存储与释放函数头文件
04  void shuru();                    //声明输入航班函数
05  void shuchu();                  //声明输出航班函数
06  void baocun();                  //声明保存航班函数
07  void duqu();                    //声明读取航班函数
08  void chazhao();                 //声明查找航班函数
09  void shanchu();                 //声明删除航班函数
10  void dingpiao();                //声明订票函数
11  void tuipiao();                 //声明退票函数
12  void xiugai();                  //声明修改航班函数
```



```
13  #define Z 1000
14  struct node                                //定义结构体数组
15  {
16      int bianhao;
17      char q[20];
18      char o[20];
19      char t[20];
20      int n;
21  }s[Z];
22  int i;
23  int r=0;
24  void main()
25  {
26      int a;
27      printf("    ---欢迎进入航空管理系统---\n");
28      printf("=====\\n");
29      do
30      {
31          printf(" ----- 1.输入航班模块 ----- \\n"
32              " ----- 2.输出航班模块----- \\n"
33              " ----- 3.修改航班模块----- \\n"
34              " ----- 4.查找航班模块----- \\n"
35              " ----- 5.删除航班模块----- \\n"
36              " ----- 6.订票模块----- \\n"
37              " ----- 7.退票模块----- \\n"
38              " ----- 8.退出系统----- \\n");
39          printf("=====\\n");
40          printf("在 1-8 中选择相应功能: ");
41          scanf("%d",&a);
42          switch(a)
43          {                                //switch 结构
44              case 1: shuru();              //调用输入模块
45              break;
46              case 2: shuchu();             //调用输出模块
47              break;
48              case 3: xiugai();             //调用修改模块
49              break;
50              case 4: chazhao();            //调用查找模块
51              break;
52              case 5: shanchu();            //调用删除模块
53              break;
54              case 6: dingpiao();           //调用订票模块
55              break;
56              case 7: tui piao();           //调用退票模块
57              break;
58              case 8:;
```




```
59     break;
60     }
61     }while(a!=8);           //判断结束
62     }
63     void shuru()           //输出模块函数
64     {
65         int z;
66         printf("请输入航班信息:\n");
67
68     printf("-----\n");
69     for(i=0;i<Z;i++)
70     {
71         printf("输入航班编号:\n");
72         scanf("%d",&s[i].bianhao);           //输入航班号
73         printf("输入航班起点:\n");
74         scanf("%s",s[i].q);                 //输入起始站
75         printf("输入航班目的地:\n");
76         scanf("%s",s[i].o);                 //输入终点站
77         printf("输入航班所需时间:\n");
78         scanf("%s",s[i].t);                 //输入时间
79         printf("输入航班的机票数:\n");
80         scanf("%d",&s[i].n);                 //输入机票数
81         r++;
82         printf("第%d 个航班信息已添加完,是否继续?按 0 结束,按其他键继续\n",r);
83         scanf("%d",&z);
84         if(z==0)
85         {
86             baocun();           //将结构体信息存盘
87             shuchu();           //输出、输入的航班信息
88             break;
89         }
90     }
91     void baocun()               //保存模块程序
92     {
93         FILE *fp,*fpl;
94         int i;                 //定义文件指针
95         if((fp=fopen("j.txt","w"))==NULL)           //打开文件并判断是否出错
96         {
97             printf("打开文件失败!\n\n");           //打印出错提示
98         }
99         if((fpl=fopen("x.txt","w"))==NULL)           //打开文件并判断是否出错
100        {
101            printf("打开文件失败!\n\n");           //打印出错提示
102        }
```



```
103     for(i=0;i<r;i++)
104     if(!fwrite(&s[i],sizeof(struct node),1,fp)) //向文件写入数据，并判断是否出错
105     printf("写入失败!\n\n");
106     fprintf(fp1,"%d",r);
107     fclose(fp); //关闭文件指针 fp
108     fclose(fp1); //关闭文件指针 fp1
109 }
110 void duqu() //从文件读取信息模块
111 {
112     FILE *fp,*fp1;int i; //定义文件指针
113     if((fp=fopen("j.txt","r"))==NULL) //打开文件并判断是否出错
114     {
115         printf("打开文件出错\n"); //打印出错提示
116     }
117     if((fp1=fopen("x.txt","r"))==NULL) //打开文件并判断是否出错
118     {
119         printf("打开文件出错!\n\n"); //打印出错提示
120     }
121     fscanf(fp1,"%d",&r);
122     for(i=0;i<r;i++)
123     {
124         fread(&s[i],sizeof(struct node),1,fp); //从文件中读取信息
125     }
126     fclose(fp);
127     fclose(fp1); //关闭文件
128 }
129 void shuchu() //输出模块
130 {
131     duqu(); //调用读取文件函数
132     printf("航班编号 起点 目的地 花费时间 剩余机票数\n");
133     for(i=0;i<r;i++)
134     {
135         printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n); //输出航班信息
136     }
137 }
138 void chazhao() //查询模块
139 {
140     char n1[20];
141     char n2[20];
142     char it[10];
143     int t,t1;
144     do
145     {
146         printf("请选择航班查找的方式: \n"); //打印查询方式菜单
147         printf("1.通过航班编号查找\n\n")
```



```
148     "2.通过目的地查找\n\n"
149     "3.通过航班线路查找\n\n"
150     "4.返回\n\n");
151     printf("在 1-4 中选择: \n");
152     scanf("%d",&t);                //读取查找方式
153     if(t==4)
154         break;
155     switch(t)
156     {
157     case 1:
158         printf("输入要查询的航班编号: \n");
159         scanf("%d",t1);            //从键盘输入航班号
160         break;
161     case 2:
162         printf("输入目的地: \n");
163         scanf("%s",n2);//读取终点站
164         break;
165     case 3:
166         printf("输入起点: \n");
167         scanf("%s",n1);//读取起始站
168         printf("输入目的地: \n");
169         scanf("%s",n2);//终点站
170         break;
171     }
172     duqu();                        //调用读取函数
173     for(i=0;i<r;i++)
174     {
175         if(strcmp(s[i].q,n1)==0||strcmp(s[i].o,n2)==0)
176         {
177             printf("\n 查找到相应航班!\n");
178             printf("航班号 起点 目的地 花费时间 剩余机票数\n");
179             printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n);
180                                     //输出信息
181             break;
182         }
183         if(s[i].bianhao==t1)        //按航班号判断输出条件
184         {
185             printf("\n 查找到相应航班!\n");
186             printf("航班号 起始站 终点站 时间 机票数\n");
187             printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n);
188                                     //输出信息
189             break;
190         }
191     }
192     t1=0;//将航班号赋值为 0
193     printf("是否继续查找航班?键入 yes 继续, 其他键结束\n");
```



```
192     scanf("%s",it);
193 }while(strcmp(it,"yes")==0);           //判断结束
194 }
195 void shanchu()                         //删除模块
196 {
197     char n1[20];
198     char n2[20];
199     char it[20];
200     int t,t1;
201     do
202     {
203         printf("请输入航班的删除方式: \n");           //打印删除方式菜单
204         printf("1.通过航班编号删除航班\n"
205             "2.通过航线删除航班\n"
206             "3.返回\n\n");
207         printf("请在 1-3 中选择以回车键结束: \n");
208         scanf("%d",&t);                               //读取删除方式
209         if(t==3)
210             break;                                     //跳出循环
211         switch(t)
212         {
213             case 1:
214                 printf("请输入航班编号: \n");
215                 scanf("%d",&t1);                       //读取航班号
216                 duqu();                                   //调用读取函数
217                 break;                                   //跳出循环
218             case 2:
219                 printf("请输入起点: \n");
220                 scanf("%s",n1);                           //读取起始站
221                 printf("请输入目的地: \n");
222                 scanf("%s",n2);                           //读取终点站
223                 duqu();                                   //调用读取函数
224                 break;                                   //跳出循环
225             }
226             for(i=0;i<r;i++)
227             {
228                 if(s[i].bianhao==t1|| (strcmp(s[i].q,n1)==0&&strcmp(s[i].o,n2)==0))
229                     //判断输入信息是否存在
230                 {
231                     s[i]=s[r-1];
232                     r--;
233                 }
234             }
235             printf("是否继续删除\n");
236             printf("输入 yes 继续, 其他键退出\n");
237             scanf("%s",it);                               //读取是否继续信息
```



```
237     baocun(); //调用读取函数
238     }while(strcmp(it,"yes")==0); //判断结束
239 }
240 void dingpiao() //订票模块
241 {
242     int p;
243     char w[10];
244     do
245     {
246         chazhao(); //调用查询模块
247         printf("输入要订的机票数: \n");
248         scanf("%d",&p); //输入所订机票数
249         if(p<0)
250         {
251             printf("请输入正确的机票数!\n"); //判断机票数是否出错
252             break;
253         }
254         if(s[i].n!=0&& s[i].n>=p) //判断是否出错
255         {
256             s[i].n=s[i].n-p;
257             baocun(); //调用保存函数
258             printf("订票成功!\n");
259             break;
260         }
261         else if(s[i].n<p) //判断机票数是否充足
262         {
263             printf("机票余数不足!\n");
264             break;
265         }
266         printf("是否继续? 输入 yes 继续, 其他键结束:\n"); //提示是否继续订票
267         scanf("%s",w);
268         }while(strcmp(w,"yes")==0); //判断结束
269     }
270 void tuipiao() //退票模块
271 {
272     int p;
273     char w[10];
274     do
275     {
276         chazhao(); //调用查询模块
277         printf("输入要退的机票数: \n");
278         scanf("%d",&p); //输入所退票数
279         if(p<0) //判断票数是否有效
280             printf("请输入正确的机票数! \n");
281         if(p>0)
282             s[i].n=s[i].n+p;
```



```
283     baocun(); //调用保存模块
284     printf("退票成功!\n");
285     printf("是否继续退票? 输入 yes 继续, 其他键结束:\n"); //判断是否继续退票
286     scanf("%s",w);
287 }while(strcmp(w,"yes")==0); //判断并跳出循环
288 }
289 void xiugai() //修改函数
290 {
291     struct node1 //定义结构体
292     {
293         int num;
294         char n1[20];
295         char n2[20];
296         char t[20];
297         int sum;
298     }q;
299     int x;
300     char w[10];
301     duqu(); //调用读取模块
302     do
303     {
304         printf("选择修改航班的方式: \n"
305             "1,通过航班编号修改:\n"
306             "2,通过航班线路修改: \n");
307         printf("请在 1--2 中选择修改方式: \n\n");
308         scanf("%d",&x); //读取修改方式
309         switch(x)
310         {
311             case 1:printf("请输入航班编号: \n");
312                 scanf("%d",&q.num); //读取航班号
313                 break;
314             case 2:printf("请输入起点: \n");
315                 scanf("%s",q.n1); //读取起始站
316                 printf("请输入目的地: \n");
317                 scanf("%s",q.n2); //读取终点站
318                 break;
319         }
320         for(i=0;i<r;i++)
321         {
322             if(strcmp(s[i].q,q.n1)==0&&strcmp(s[i].o,q.n2)==0) //判断输出条件
323             {
324                 printf("航班编号 起点 目的地 花费时间 剩余机票数\n");
325                 printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n);
326                 break;
327             }
328             if(s[i].bianhao==q.num) //判断输出条件
```



```
329     {
330         printf("航班编号 起点 目的地 花费时间 剩余机票数\n");
331         printf("%d %s %s %s %d",s[i].bianhao,s[i].q,s[i].o,s[i].t,s[i].n);
332         break;
333     }
334 }
335 q.num=0; //将结构体中的号置为零
336 printf("请输入新的航班编号、起点、目的地、时间、机票数: \n");
337 scanf("%d%s%s%d",&q.num,q.n1,q.n2,q.t,&q.sum); //输入航班
338 s[i].bianhao=q.num; //替换航班号
339 strcpy(s[i].q,q.n1); //替换其始站
340 strcpy(s[i].o,q.n2); //替换终点站
341 strcpy(s[i].t,q.t); //替换时间
342 s[i].n=q.sum; //替换机票数
343 baocun(); //调用保存模块
344 printf("修改成功! 是否继续? 键入 yes 继续, 其他键结束:\n");
345 scanf("%s",w);
346 }while(strcmp(w,"yes")==0); //判断结束
347 }
```

【代码分析】本例为飞机票管理系统范例，详细代码分析如下：

- 第 1~12 行，通过#include 命令包含相应的头文件以及自定义函数，其中 4~12 行声明用户自定义的函数。
- 第 13 行，定义了一个宏 Z，其值为 1000。
- 第 14~21 行，定义结构体数组 s，用来存储航班编号、起始点、目的地、所需时间、票数等多种类型数据。
- 第 29~61 行，通过 do-while 循环和 switch 结构实现菜单的功能，键入不同的数字调用相应的函数从而实现特定的功能。
- 第 63~90 行为 shuru()函数，用来实现航班信息的输入。
- 第 70~79 行，将键盘输入的数据保存至相应的结构体数组中。
- 第 91~109 行为自定义函数 baocun()，用来将航班信息保存至文件中。
- 第 110~128 行，自定义函数 duqu()，用来读取航班信息。
- 第 113~120 行，通过文件指针以读的方式打开相应的文件。
- 第 121 行，利用 fscanf()函数读取航班个数至变量 r 中。
- 第 122~125 行，从 fp 所指文件读取航班信息保存至结构体数组 s 中。
- 第 126、127 行，关闭文件指针 fp 和 fp1。
- 第 129~137 行，自定义函数 shuchu()，用来显示航班信息至屏幕。
- 第 155~171 行为 switch 结构。从键盘输入不同的数字，执行不同的 case 事件。
- 第 175~181 行，通过航班的起点和目的地来查找航班信息。
- 第 182~188 行，通过航班编号来查找航班的信息。
- 第 195~239 行，自定义函数 shanchu()。该函数用来删除特定的航班信息，其中要删除的航班由键盘输入。



- 第 211~225 行为 switch 结构。从键盘输入不同的数字，通过不同的方式删除航班信息。
- 第 226~233 行，将结构体最后一个元素赋给 s[i]，从而实现第 i 个航班的删除功能。
- 第 240~269 行，自定义函数 dingpiao()。该函数用来实现订票功能，票数和航班编号由键盘输入。
- 第 254~260 行，若用户输入的票数大于 0 且小于该航班的剩余票数则订票成功，否则失败。
- 第 270~288 行，自定义函数 tuipiao()。该函数用来实现退票的功能，退票的航班及票数由键盘输入。
- 第 281~284 行，若用户输入的票数大于 0 则退票成功。
- 第 289~347 行，自定义函数 xiugai()。该函数用来修改航班信息然后写至相应的文件中，其中新的航班信息由键盘输入。

【运行结果】该程序的执行结果如图 17-2 所示。

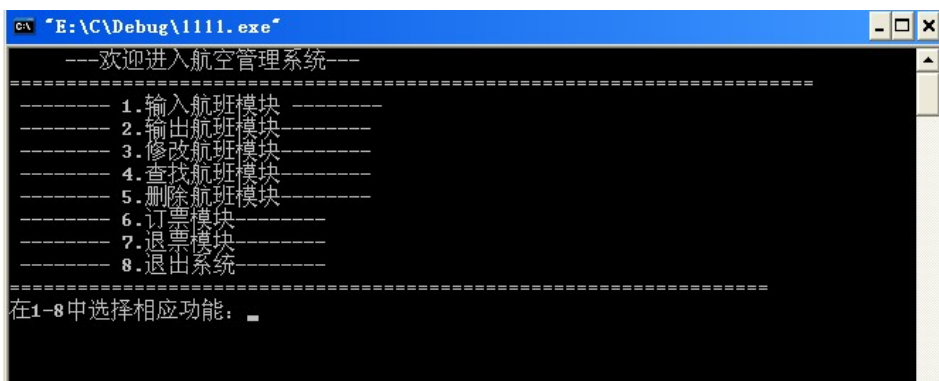


图 17-2 范例 17.1 结果图



17.6 小结

本章结合 C 语言知识开发了一个航空订票管理系统，该系统包含查询航班、添加和删除航班、订票、退票等功能，方便机场工作人员对航班信息的管理。在开发的过程中，主要利用了 do-while 循环、for 循环及 C 语言的基本输入、输出函数等实现相应的功能。读者可认真学习和分析本章的代码，结合 C 语言所学知识开发其他的系统。

第 18 章 C 语言开发项目： 学生管理系统

在第 17 章中讲解了如何利用 C 语言开发航空订票管理系统，在本章中将进一步讲解如何利用 C 语言进行学生管理系统的开发。

本章主要涉及的知识点有：

- 学生资料系统的分析；
- 学生资料变量的定义；
- 学生资料系统设计。



18.1 学生管理系统需求分析

随着科学技术的不断发展，计算机科学逐渐成熟，计算机强大的功能已被人们深刻认识，它已进入社会的各个领域并发挥着至关重要的作用。如今，学校学生管理是教务管理中一个最重要的问题，它是整个学校管理的核心。随着学校规模的不断扩大，学生人数的每年上升，学生的管理也变得越来越繁杂。

以前学校一直使用传统人工的方法来管理文件档案、统计和查询学生资料，这种管理方式存在着许多缺点。例如工作效率低、保密性低等，另外时间越长，文件和数据会堆积得越来越多。因此，开发一个学生管理系统用来管理学生资料，对于促进学校学生的管理和提高学校教学质量与办学水平有着显著意义。

作为计算机应用的一部分，使用计算机对学生资料进行管理，能够极大地提高学生资料管理的效率。因此，学生管理系统的设计与开发也是一个时代必经的阶段。



18.2 学生管理系统总体设计

学生信息包括学号、姓名、性别、年龄、籍贯、成绩、考勤等数据，其实现代码如下：

```
struct st //定义结构数组,存储学员信息
{
    int bianhao; //学号
    char n[20]; //姓名
    char sex; //性别
    int age; //年龄
    char jiguan[20]; //籍贯
    float fenshu; //成绩
    int kaoqing; //考勤
```



```
char num[20]; //联系电话
char ad[50]; //地址
}s;
```

学生信息结构图如图 18-1 所示。

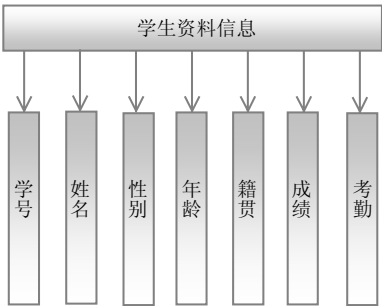


图 18-1 学生资料结构图

18.2.1 添加学生资料

学生资料添加代码如下所示。

```
printf("输入要添加的学员资料:\n");
printf("输入学员编号:");
scanf("%d",&s[x].bianhao); //输入学生编号
printf("输入学员姓名:");
scanf("%s",s[x].n); //输入学生姓名
printf("输入学员性别:");
c=getchar(); //输入学生性别
s[x].sex=c;
printf("输入学员年龄:");
scanf("%d",&s[x].age); //输入学生年龄
printf("输入学员籍贯:");
gets(s[x].jiguan); //输入学生籍贯
printf("输入学员分数:");
scanf("%f",&s[x].fenshu); //输入学生分数
printf("输入学员考勤次数:");
scanf("%d",&s[x].kaoqing); //输入学生考勤次数
printf("输入学员电话:");
gets(s[x].num); //输入学生电话
printf("输入学员住址:");
gets(s[x].adr); //输入学生住址
```

18.2.2 查询学生资料

学生资料查询代码如下所示。

(1) 通过姓名查询学生资料。

```
printf("输入要查询的学员姓名: \n");
```



```
scanf("%s",n1); //输入要查询的学生姓名
for(x=0;x<N;x++)
{
    if(!strcmp(s[x].n,n1)) //将输入的学生姓名与s[x]中的姓名逐个比较
    {
        shuchul();
        break;
    }
}
```

(2) 通过学号查询学生资料。

```
printf("输入要查询的学生编号:");
scanf("%d",&y); //输入学生编号
for(x=0;x<N;x++)
{
    if(y==s[x].bianhao) //将输入的学生编号与s[x]中的编号逐个比较
    {
        shuchul(); //调用输出函数输出信息
        break;
    }
}
```

18.2.3 排序学生资料

学生资料排序代码如下所示。

(1) 通过学生考勤次数排序。

```
for(j=0;j<N-x-1;j++)
{
    if(s[x].bianhao==0) //判断s[x]记录是否为空
    {
        break;
    }
    if(s[j].kaoqing<stu[j+1].kaoqing) //根据考勤次数排序学生资料
    {
        s1=s[j];
        s[j]=s[j+1];
        s[j+1]=s1;
    }
}
```

(2) 通过学生分数排序。

```
for(j=0;j<N-x-1;j++)
{
    if(s[x].bianhao==0) //判断s[x]记录是否为空
    {
```



```

        break;
    }
    if(s[j].fenshu<s[j+1].fenshu)           //根据分数排序学生资料
    {
        s1=s[j];
        s[j]=s[j+1];
        s[j+1]=s1;
    }
}

```

(3) 通过学生性别排序。

```

for(x=0;x<N-1;x++)
{
    if(s[x].bianhao==0)                    //判断s[x]记录是否为空
        break;
    if(s[x].sex=='m' || s[x].sex=='M')      //输出男生资料
    {
        printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
            s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
        printf("\n");
    }
}
for(x=0;x<N-1;x++)
{
    if(s[x].sex=='W' || s[x].sex=='w')      //输出女生资料
    {
        printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
            s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
        printf("\n");
    }
}

```

18.2.4 删除学生资料

学生资料删除代码如下所示。

```

printf("输入要删除第几个学员资料:");
scanf("%d",&t1);                          //输入要删除的学员位置
printf("确定删除吗?输入 Y 确认, 输入 N 退出 ");
scanf("%c",&c);                            //确定是否删除
printf("\n 编号\t 姓名\t 性别\t 年龄\t 籍贯\t 成绩\t 考勤次数\t 电话号码\t 住址\n");
if(c=='y' || c=='Y')
{
    for(x=t1-1;x<t;x++)                    //将要删除的结构体后面的元素都向前移动一位
    {

```



```
s[x]=s[x+1];
}
for(x=0;x<t-1;x++)
{
    if(s[x].bianhao==0)                //判断 s[x] 记录是否为空
    {
        break;
    }
    printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
        s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
}
t=t-1;
}
```

18.2.5 修改学生资料

学生资料修改代码如下所示。

```
printf("输入要修改的学员资料:\n");
printf("输入新的姓名:");
scanf("%s",n1);                        //输入学生编号
strcpy(s[x].n,n1);                     //替换学生编号
printf("输入新的性别:");
c=getchar();                           //输入学生性别
s[x].sex=c;
printf("输入新的成绩:");
scanf("%f",&s);                        //输入学生分数
s[x].fenshu=s;                          //替换学生的分数
printf("输入新的年龄:");
scanf("%d",&age);                      //输入学生年龄
s[x].age=age;                           //替换学生年龄
printf("输入新的籍贯:");
gets(ji);                              //输入学生籍贯
strcpy(s[x].jiguan,ji);                 //替换学生籍贯
printf("输入新的考勤:");
scanf("%d",&k);                        //输入学生考勤次数
s[x].kaoqing=k;                        //替换学生考勤
printf("输入新的电话:");
gets(num);                             //输入学生电话
strcpy(s[x].num,num);                  //替换学生住址
printf("输入新的地址:");
gets(adr);                             //输入学生住址
strcpy(s[x].ad,adr);                   //替换学生住址
```



18.3 学生管理系统程序实现

【范例 18.1】利用 C 语言，设计一个学生管理系统，能对学生资料进行管理，要求具有以下功能：

- (1) 添加学生资料。
- (2) 根据姓名、学号查询学生信息。
- (3) 删除学生的资料。
- (4) 根据姓名、学号等修改学生资料。

分析：将学生信息定义为结构体类型，其中包括学号、姓名、性别、年龄、籍贯、成绩、考勤等数据，然后将添加学生资料、删除学生资料等分别编写成不同的函数保存在不同的文件中，最后在主函数中调用函数实现相应的功能。

范例 18.1 代码实现

main.cpp 文件内容如下：

```

01  #include <stdio.h>                //包含 stdio.h 头文件
02  #include <windows.h>              //包含 windows.h 头文件
03  #include <string.h>                //包含 string.h 头文件
04  #include "print1.h"                //包含自定义文件 print1.cpp
05  #include "print.h"                 //包含自定义文件 print.cpp
06  #include "add.h"                   //包含自定义文件 add.cpp
07  #include "sortf.h"                 //包含自定义文件 sortf.cpp
08  #include "sorts.h"                 //包含自定义文件 sorts.cpp
09  #include "sort.h"                  //包含自定义文件 sort.cpp
10  #include "search.h"                //包含自定义文件 search.cpp
11  #include "searchbynum.h"           //包含自定义文件 searchbynum.cpp
12  #include "searchbyname.h"          //包含自定义文件 searchbyname.cpp
13  #include "searchsingle.h"          //包含自定义文件 searchsingle.cpp
14  #include "insert.h"                //包含自定义文件 shanchu.cpp
15  #include "shanchu.h"               //包含自定义文件 search.cpp
16  #include "changebyname.h"          //包含自定义文件 insert.cpp
17  #include "changebynum.h"           //包含自定义文件 changebyname.cpp
18  #include "xiugai.h"                //包含自定义文件 xiugai.cpp
19  #define N 50                       //定义宏常量
20  struct st sl;
21  int x;
22  void main()
23  {
24      int m;
25      printf("-----欢迎进入学生管理系统 -----\n");
26      printf("*****\n");

```



```
27  printf("  1  添加学生资料模块 \n");
28  printf("  2  查找学生资料模块 \n");
29  printf("  3  修改学生资料模块 \n");
30  printf("  4  删除学生资料模块 \n");
31  printf("  5  退出      \n");
32  printf("*****\n");
33  printf("输入要进入的系统模块:");
34  scanf("%d",&m);
35  if(m>5||m<1)                                //判断用户输入的选项是否符合要求
36  {
37      printf("输入有误, 请输入 1~5 之间的数字\n");
38  }
39  else
40  {
41      switch(m)                                //switch 结构
42      {
43          case 1:tianjia();                    //添加学员资料
44          break;
45          case 2:chazhao();                    //查询学员资料
46          break;
47          case 3:xiugai();                     //修改学员资料
48          break;
49          case 4:shanchu();                    //删除学员资料
50          break;
51          case 5:exit(1);                      //退出系统
52          break;
53      }
54  }
55  }
```

print1.h 文本内容如下:

```
01  #include <stdio.h>
02  struct st                                //定义结构数组, 存储学员信息
03  {
04      int bianhao;                          //学号
05      char n[20];                            //姓名
06      char sex;                              //性别
07      int age;                              //年龄
08      char jiguan[20];                       //籍贯
09      float fenshu;                          //成绩
10      int kaoqing;                          //考勤
11      char num[20];                          //联系电话
12      char ad[50];                          //地址
13  }s[50]={                                  //初始化学生信息
14      {19,"李红",'W',20,"江西",45,90,"075586013388","江西省南昌市八一广场"},
15      {4,"康文明",'M',22,"广州",62.5,80,"13511007788","广州东莞南湖公寓"},
```



```

16  {7,"王吉",'W',24,"湖南",92.5,75,"13875533445","湖南长沙步行街"},
17  {39,"郭家",'M',23,"湖北",87,77,"075588889999","湖北武汉步行街大道"},
18  {10,"王宏",'M',21,"四川",58,78,"13675555667","四川成都公寓"}
19  };
20  void shuchul()                                //打印单个学生信息函数
21  {
22      int x;
23      printf("\n编号\t姓名\t性别\t年龄\t籍贯\t成绩\t考勤次数\t电话号码\t住址\n");
24      printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
25      s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
26  }

```

print.h 文本内容如下:

```

01  #include <stdio.h>
02  void shuchu()                                //打印所有成员函数
03  {
04      int x;
05      printf("\n编号\t姓名\t性别\t年龄\t籍贯\t成绩\t考勤次数\t电话号码\t住址\n");
06      for(x=0;x<50;x++)
07      {
08          if(s[x].bianhao==0)                  //判断 s[x] 记录是否为空
09          {
10              break;
11          }
12          printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
13          s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
14          printf("\n");
15      }
16  }

```

sortf.h 文本内容如下:

```

01  extern struct st s1;
02  void sortf()                                //按考勤由高到低显示学员资料
03  {
04      int temp=0,j,x;
05      for(x=0;x<50;x++)
06      {
07          if(s[x].bianhao==0)                  //判断 s[x] 记录是否为空
08          {
09              break;
10          }
11          for(j=0;j<50-x-1;j++)
12          {
13              if(s[x].bianhao==0)              //判断 s[x] 记录是否为空
14              {
15                  break;

```




由浅入深学 C 语言——基础、进阶与必做 430 题

```
16     }
17     if(s[j].kaoqing<s[j+1].kaoqing)           //根据考勤次数对学生资料排序
18     {
19         s1=s[j];
20         s[j]=s[j+1];
21         s[j+1]=s1;
22     }
23 }
24 }
25 shuchu();                                     //调用输出函数
26 }
```

sorts.h 文件内容如下:

```
01 void sorts()                                 //按成绩由高到低显示学员资料
02 {
03     int j,x;
04     for(x=0;x<50;x++)
05     {
06         if(s[x].bianhao==0)                 //判断s[x]记录是否为空
07         {
08             break;
09         }
10         for(j=0;j<50-x-1;j++)
11         {
12             if(s[x].bianhao==0)             //判断s[x]记录是否为空
13             {
14                 break;
15             }
16             if(s[j].fenshu<s[j+1].fenshu)    //根据分数对学生资料排序
17             {
18                 s1=s[j];
19                 s[j]=s[j+1];
20                 s[j+1]=s1;
21             }
22         }
23     }
24     shuchu();                               //调用输出函数
25 }
```

sort.h 文件内容如下:

```
01 void sort()                                 //按性别排列学员资料
02 {
03     int x;
04     printf("\n 编号\t 姓名\t 性别\t 年龄\t 籍贯\t 成绩\t 考勤次数\t 电话号码\t 住址\n");
05     for(x=0;x<49;x++)
06     {
```



```

07     if(s[x].bianhao==0)                //判断 s[x] 记录是否为空
08         break;
09     if(s[x].sex=='m' || s[x].sex=='M')    //输出男生资料
10     {
11         printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
12             s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
13         printf("\n");
14     }
15 }
16 for(x=0;x<49;x++)
17 {
18     if(s[x].bianhao==0)                //判断 s[x] 记录是否为空
19         break;
20     if(s[x].sex=='W' || s[x].sex=='w')    //输出女生资料
21     {
22         printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
23             s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
24         printf("\n");
25     }
26 }
27 }

```

searchbynum.h 文件内容如下：

```

01     #include <search.h>
02     void chazhao();
03     void searchnum()                    //按学号排列显示学员资料
04     {
05         int y,x;
06         char c;
07         printf("输入要查询的学生编号:");
08         scanf("%d",&y);                //输入要查询的学生编号
09         for(x=0;x<50;x++)
10         {
11             if(y==s[x].bianhao)        //将输入的学生编号与 s[x] 中的编号逐个比较
12             {
13                 shuchul();              //若相等则输出
14                 break;
15             }
16         }
17         while(1)                        //循环判断用户是否继续使用该功能
18         {
19             printf("是否继续查询学员信息?输入 Y 继续, 输入 N 结束");
20             scanf("%c",&c);              //从键盘获取字符至变量 c 中
21             if(c=='y' || c=='Y')        //若满足该条件则调用 searchnum() 函数
22             {
23                 searchnum();

```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
24     }
25     else if(c=='n' || c=='N')           //若满足该条件则调用 chazhao()函数
26     {
27         chazhao();
28     }
29 }
30 }
```

searchbyname.h 文件内容如下:

```
01 void searchname()                     //自定义函数 searchname()
02 {
03     int x;
04     char nl[20];
05     char c;
06     printf("输入要查询的学员姓名: \n");
07     scanf("%s", nl);                   //输入要查找的学生姓名
08     for(x=0; x<50; x++)
09     {
10         if(!strcmp(s[x].n, nl))        //将输入的学生姓名与s[x]中的姓名逐个比较
11         {
12             shuchul();                  //调用输出函数
13             break;
14         }
15     }
16     while(1)                           //循环判断用户是否继续使用该功能
17     {
18         printf("是否继续查询学员信息?输入 Y 继续, 输入 N 结束");
19         scanf("%c", &c);                 //从键盘获取字符至变量 c 中
20         if(c=='Y' || c=='y')             //若满足该条件则调用 searchnum()函数
21         {
22             searchnum();
23         }
24         else if(c=='n' || c=='N')        //若满足该条件则调用 chazhao()函数
25         {
26             chazhao();
27         }
28     }
29 }
```

searchsingle.h 文件内容如下:

```
01 void searchsingle()                   //自定义函数 searchsingle()
02 {
03     int y;
04     char c;
05     printf("选择要查询学员的方式:\n"); //输出提示信息
06     printf("1.通过学号排列查找学生资料\n");
```



```

07  printf("2.通过别排列查找学生资料\n");
08  printf("3.通过成绩高低查找学员资料\n");
09  printf("4.通过考勤高低查找学员资料\n");
10  printf("5.返回\n");
11  scanf("%d",&y); //从键盘获取数字至变量 y 中
12  if(y>5||y<1) //判断用户输入的选项是否符合要求
13  {
14      printf("输入有误, 输入 1~5 之间的数字\n");
15  }
16  switch(y)
17  {
18      case 1:shuchu(); //按学号排列显示学员资料
19      while(1) //循环判断用户是否继续使用该功能
20      {
21          printf("是否继续查询学员信息?输入 Y 继续, 输入 N 结束");
22          scanf("%c",&c); //从键盘获取字符至变量 c 中
23          if(c=='y' || c=='Y') //若满足该条件则调用 searchnum() 函数
24          {
25              searchnum();
26          }
27          else if(c=='n' || c=='N') //若满足该条件则调用 chazhao() 函数
28          {
29              chazhao();
30          }
31      }
32      break;
33      case 2:sort(); //按性别排列学员资料
34      break;
35      case 3:sorts(); //按成绩由高到低显示学员资料
36      break;
37      case 4:sortf(); //按考勤由高到低显示学员资料
38      break;
39      case 5:chazhao(); //返回查询主界面
40      break;
41  }
42  }

```

search.h 文件内容如下:

```

01  void searchname();
02  void searchnum();
03  void searchsingle();
04  void chazhao() //查询函数
05  {
06      int y;
07      printf("输入要查找学员的方式:\n"); //输出提示信息
08      printf("1.通过姓名查询:\n");

```



由浅入深学 C 语言——基础、进阶与必做 430 题

```
09  printf("2.通过学号查询:\n");
10  printf("3.通过成绩查询:\n");
11  printf("4.返回主界面\n");
12  scanf("%d",&y);
13  if(y>4||y<=0) //判断用户输入的选项是否符合要求
14  {
15      printf("输入有误,输入1-4之间的数字\n");
16      chazhao(); //若用户输入选项不符合要求,重新进入主界面
17  }
18  switch(y)
19  {
20      case 1:searchname(); //输入姓名查询
21      break;
22      case 2:searchnum(); //输入学号查询
23      break;
24      case 3:searchsingle(); //特殊查询方式
25      break;
26      case 4:
27      break;
28  }
29  }
```

add.h 文件内容如下:

```
01  void tianjia() //添加学生信息函数
02  {
03      int c;
04      shuchu(); //调用输出函数
05      for(x=0;x<50;x++)
06      {
07          if(s[x].bianhao==0) //判断s[x]记录是否为空
08          {
09              printf("输入要添加的学员资料:\n");
10              printf("输入学员编号:");
11              scanf("%d",&s[x].bianhao); //输入要添加的学生编号
12              printf("输入学员姓名:");
13              scanf("%s",s[x].n); //输入要添加的学生姓名
14              printf("输入学员性别:");
15              c=getchar(); //输入要添加的学生性别
16              s[x].sex=c;
17              printf("输入学员年龄:");
18              scanf("%d",&s[x].age); //输入要添加的学生年龄
19              printf("输入学员籍贯:");
20              gets(s[x].jiguan); //输入要添加的学生籍贯
21              printf("输入学员分数:");
22              scanf("%f",&s[x].fenshu); //输入要添加的学生分数
23              printf("输入学员考勤次数:");
```



```

24     scanf("%d",&s[x].kaoqing);           //输入要添加的学生考勤次数
25     printf("输入学员电话:");
26     gets(s[x].num);                       //输入要添加的学生电话
27     printf("输入学员住址:");
28     gets(s[x].adr);                       //输入要添加的学生住址
29     break;
30 }
31 }
32 shuchu();
33 while(1)                                  //循环判断用户是否继续使用该功能
34 {
35     printf("是否继续添加学员信息?输入 Y 继续, 输入 N 结束");
36     scanf("%c",&c);                       //从键盘获取字符至变量 c 中
37     if(c=='Y' || c=='y')                  //若满足该条件则调用 tianjia() 函数
38     {
39         tianjia();
40     }
41     else if(c=='n' || c=='N')              //若满足该条件则退出循环
42     break;
43 }
44 }

```

insert.h 文件内容如下:

```

01 void charu()                             //插入学员资料函数
02 {
03     int i,x;
04     char c;
05     printf("输入要插入学员资料:\n");
06     printf("输入该学员编号:");
07     scanf("%d",&s1.bianhao);              //输入学生编号
08     printf("输入学员姓名:");
09     scanf("%s",s1.n);                     //输入学生姓名
10     printf("输入学员性别:");
11     scanf("%c",&s1.sex);                  //输入学生性别
12     printf("输入学员年龄:");
13     scanf("%d",&s1.age);                  //输入学生年龄
14     printf("输入学员籍贯:");
15     scanf("%s",s1.jiguan);                //输入学生籍贯
16     printf("输入学员成绩:");
17     scanf("%f",&s1.fenshu);              //输入学生分数
18     printf("输入学员考勤次数:");
19     scanf("%d",&s1.kaoqing);             //输入学生考勤次数
20     printf("输入学员号码:");
21     scanf("%s",s1.num);                   //输入学生号码
22     printf("输入学员住址:");
23     scanf("%s",s1.ad);                    //输入学生住址

```



```
24     for(x=0;x<50;x++)
25     {
26         if(s1.bianhao>s[x].bianhao)                //通过学生编号进行比较
27         {
28             for(i=0;i<50;i++)
29             if(s[i].bianhao==0)                    //若 s[j] 记录为空
30             {
31                 break;
32             }
33         }
34         for(;i>x+1;i--)                            //将数组中的元素都向前移动一位
35         {
36             s[i]=s[i-1];
37         }
38         s[x+1]=s1;                                //将元素插入数组中
39     }
40     while(1)                                       //循环判断用户是否继续使用该功能
41     {
42         printf("是否继续插入学员信息?输入 Y 继续, 输入 N 结束");
43         scanf("%c",&c);                            //从键盘获取字符至变量 c 中
44         if(c=='y' || c=='Y')                      //若满足该条件则调用 charu() 函数
45         {
46             charu();
47         }
48         else if(c=='n' || c=='N')                 //若满足该条件则退出循环
49             break;
50     }
51 }
```

changebyname.h 文件内容如下:

```
01     void changebyname()                          //输入姓名开始修改
02     {
03         int x;
04         char n1[20],c;
05         float f;
06         int age;                                  //年龄
07         char ji[20];                              //籍贯
08         int k;                                    //考勤
09         char num[15];                             //联系电话
10         char adr[30];                             //地址
11         shuchu();
12         printf("输入要修改的学员姓名:");
13         scanf("%s",n1);                            //输入要修改的学生姓名
14         for(x=0;x<10;x++)
15         {
16             if(strcmp(n1,s[x].n)==0)              //将输入的学生姓名与 s[x] 中的姓名逐个比较
```



```

17     {
18         shuchul(); //若相等则输出
19         break;
20     }
21 }
22 printf("输入要修改的学员资料:\n");
23 printf("输入新的姓名:");
24 scanf("%s",n1); //输入新的学生姓名
25 strcpy(s[x].n,n1); //替换原来的学生姓名
26 printf("输入新的性别:");
27 c=getchar(); //输入新的学生性别
28 s[x].sex=c; //替换原来的学生性别
29 printf("输入新的成绩:");
30 scanf("%f",&f); //输入新的学生分数
31 s[x].fenshu=f; //替换原来的学生分数
32 printf("输入新的年龄:");
33 scanf("%d",&age); //输入新的学生年龄
34 s[x].age=age; //替换原来的学生年龄
35 printf("输入新的籍贯:");
36 gets(ji); //输入新的学生籍贯
37 strcpy(s[x].jiguan,ji); //替换原来的学生籍贯
38 printf("输入新的考勤:");
39 scanf("%d",&k); //输入新的学生考勤次数
40 s[x].kaoqing=k; //替换原来的学生考勤
41 printf("输入新的电话:");
42 gets(num); //输入新的学生电话
43 strcpy(s[x].num,num); //替换原来的学生电话
44 printf("输入新的地址:");
45 gets(adr); //输入新的学生住址
46 strcpy(s[x].ad,adr); //替换原来的学生地址
47 shuchul();
48 while(1) //循环判断用户是否继续使用该功能
49 {
50     printf("是否继续修改学员信息?输入 Y 继续, 输入 N 结束");
51     scanf("%c",&c); //从键盘获取字符至变量 c 中
52     if(c=='Y' || c=='y') //若满足该条件则调用 changebyname() 函数
53     {
54         changebyname();
55     }
56     else if(c=='n' || c=='N') //若满足该条件则退出循环
57         break;
58 }
59 }

```

changebynum.h 文件内容如下:

```

01 void changebynum() //通过编号修改函数

```




```
02  {
03     int p,q,r,x;
04  char c;
05  struct st                                //定义结构数组,存储学员信息
06  {
07     int bianhao;                          //学号
08     char n[20];                          //姓名
09     char sex;                             //性别
10     int age;                             //年龄
11     char jiguan[20];                     //籍贯
12     float fenshu;                       //成绩
13     int kaoqing;                         //考勤
14     char num[20];                       //联系电话
15     char ad[50];                        //地址
16  }t;
17  shuchu();
18  printf("输入要修改的学员学号:");
19  scanf("%d",&p);                          //输入要修改的学生编号
20  if(p<1||p>r)                             //输入错误后,重新进入修改界面
21  {
22     printf("输入有误,重新输入!");
23     changebynum();                        //调用 changebyname() 函数
24  }
25  printf("输入要修改的学员资料:\n");
26  printf("输入新的姓名:");
27  scanf("%s",t.n);                        //输入新的学生姓名
28  strcpy(s[x].n,t.n);                    //替换原来的学生姓名
29  printf("输入新的性别:");
30  t.sex=getchar();                       //输入新的学生性别
31  s[x].sex=t.sex;                        //替换原来的学生性别
32  printf("输入新的成绩:");
33  scanf("%f",&t.fenshu);                   //输入新的学生成绩
34  s[x].fenshu=t.fenshu;                  //替换原来的学生成绩
35  printf("输入新的年龄:");
36  scanf("%d",&t.age);                     //输入新的学生年龄
37  s[x].age=t.age;                       //替换原来学生年龄
38  printf("输入新的籍贯:");
39  gets(t.jiguan);                       //输入新的学生籍贯
40  strcpy(s[x].jiguan,t.jiguan);          //替换原来学生籍贯
41  printf("输入新的考勤:");
42  scanf("%d",&t.kaoqing);                 //输入新的学生考勤次数
43  s[x].kaoqing=t.kaoqing;               //替换原来学生考勤
44  printf("输入新的联系电话:");
45  gets(t.num);                          //输入新的学生电话
46  strcpy(s[x].num,t.num);               //替换原来的学生电话
47  printf("输入新的地址:");
```



```

48     gets(t.ad);                                //输入新的学生住址
49     strcpy(s[x].ad,t.ad);                      //替换原来的学生地址
50     while(1)                                    //循环判断用户是否继续使用该功能
51     {
52         printf("是否继续修改学员信息?输入 Y 继续, 输入 N 结束");
53         scanf("%c",&c);                        //从键盘获取字符至变量 c 中
54         if(c=='Y' || c=='Y')                    //若满足该条件则调用 changebyname() 函数
55         {
56             changebyname();
57         }
58         else if(c=='n' || c=='N')                //若满足该条件则退出循环
59             break;
60     }
61 }

```

xiugai.h 文件内容如下:

```

01 void xiugai()                                //修改函数
02 {
03     int c;
04     printf("输入修改学员资料的方式:\n");        //输出提示信息
05     printf("1.通过姓名查找修改\n");
06     printf("2.通过学号查找修改\n");
07     printf("3.插入学员资料\n");
08     printf("4.返回\n");
09     scanf("%d",&c);
10     if(c>4 || c<1)                            //判断用户输入的选项是否符合要求
11     {
12         printf("输入有误,请输入 1-4 之间的数字\n\n");
13         xiugai();                              //若用户输入的选项不符合要求,重新进入主界面
14     }
15     switch(c)
16     {
17         case 1:changebyname();                  //调用 changebyname() 函数
18         break;
19         case 2:changebynum();                    //调用 changebynum() 函数
20         break;
21         case 3:charu();                          //调用 charu() 函数
22         break;
23         case 4:
24         break;
25     }
26 }

```

shanchu.h 文件内容如下:

```

01 void shanchu()                                //删除函数
02 {

```



```
03  shuchu();
04  char c;                                //接收是否确定
05  int t,t1,x;                            //接收删除学员学号和计数器
06  t=50;
07  printf("输入要删除第几个学员资料:");
08  scanf("%d",&t1);                        //输入要删除的学生位置
09  printf("确定删除吗?输入 Y 确认, 输入 N 退出 ");
10  scanf("%c",&c);                        //确定是否删除
11  printf("\n 编号\t 姓名\t 性别\t 年龄\t 籍贯\t 成绩\t 考勤次数\t 电话号码\t 住址\n");
12  if(c=='Y' || c=='Y')
13  {
14      for(x=t1-1;x<t;x++)                //将要删除的学员后的学生资料都向前移动一位
15      {
16          s[x]=s[x+1];
17      }
18      for(x=0;x<t-1;x++)
19      {
20          if(s[x].bianhao==0)            //判断 s[x] 记录是否为空
21          {
22              break;
23          }
24          printf("\n%d %s %c %d %s %.2f %d %s %s \n",s[x].bianhao,s[x].n,
25              s[x].sex,s[x].age,s[x].jiguan,s[x].fenshu,s[x].kaoqing,s[x].num,s[x].ad);
26      }
27      t=t-1;
28  }
29  else if(c=='n' || c=='N')              //若满足该条件则执行 shuchu() 函数
30  {
31      shuchu();
32  }
33  else
34  {
35      printf("输入有误!");
36  }
37  while(1)                              //循环判断用户是否继续使用该功能
38  {
39      printf("是否继续删除学员信息?输入 Y 继续, 输入 N 结束");
40      scanf("%c",&c);                    //从键盘获取字符至变量 c 中
41      if(c=='Y' || c=='Y')              //若满足该条件则调用 shanchu() 函数
42      {
43          shanchu();
44      }
45      else if(c=='n' || c=='N')          //若满足该条件则退出循环
46      break;
47  }
48  }
```



【代码分析】本例为学生管理系统范例，详细代码分析如下所示。

main.cpp 文件中：

- 第 1~18 行，通过#include 命令包含库文件及自定义文件。
- 第 25~33 行，通过 printf()函数在屏幕上打印出简单的菜单。
- 第 41~54 行，通过 switch 结构执行不同的函数实现相应的功能。

print1.h 文件中：

- 第 2~19 行，定义结构体数组 s 并进行初始化。
- 第 23~25 行，输出所有学生的资料至屏幕。

sortf.h 文件中：

- 第 11~23 行，通过冒泡排序法的思想根据学生考勤对学生资料进行从小到大排序。若后面的学生考勤数大于前面的学生考勤数，则交换这两个学生的信息。
- 第 25 行，调用 shuchu()函数输出所有学生资料。

sorts.h 文件中：

- 第 10~23 行，通过冒泡排序法的思想根据学生的分数对学生资料进行从小到大排序。若后面的学生分数大于前面的学生分数，则交换这两个学生的信息。
- 第 24 行，调用 shuchu()函数输出已排好序的所有学生资料。

sort.h 文件中：

- 第 9~25 行，按学生性别进行排序并输出至屏幕。

searchbynum.h 文件中：

- 第 9~16 行，按学号查询学生资料并调用 shuchu1()函数输出至屏幕。
- 第 17~29 行，判断是否继续进行查找，若输入 Y 则继续查找，否则停止查找。

searchbyname.h 文件中：

- 第 8~15 行，按学生姓名查找学生资料并调用 shuchu1()函数输出至屏幕。
- 第 16~28 行，为 while 循环。若输入 Y 则继续通过姓名查找学生资料，否则退出循环。

add.h 文件中：

- 第 9~28 行，添加学生资料至结构体数组 s 中。
- 第 33~43，为 while 循环。若输入 Y 则继续添加学生资料，否则退出循环。

search.h 文件中：

- 第 18~28 行，为 switch 结构。该结构从键盘获取 1~3 之间不同的数字，调用不同的函数查找出学生资料。

shanchu.h 文件中：

- 第 12~17 行，将要删除的结构体后的结构体变量都向前移动一个位置，从而实现学生信息的删除功能。
- 第 37~47 行，为 while 循环。若输入 Y 则继续删除学生资料，否则退出循环。

insert.h 文件中：

- 第 6~23 行，从键盘输入数据，保存至相应的结构体变量中。
- 第 24~41 行，将结构体变量插入结构体数组中的对应位置。

changebyname.h 文件中：

- 第 13 行，从键盘获取要修改的学生姓名并保存至数组 n1 中。



- 第 22~46 行，将键盘输入的数据保存至相应的结构体变量中。

changebynum.h 文件中：

- 第 19 行，从键盘获取要修改的学生学号并保存至变量 p 中。
- 第 26~49 行，将键盘输入的数据保存至相应的结构体变量中。

xiugai.h 文件中：

- 第 15~25 行，为 switch 结构。若从键盘输入 1，则执行 changebyname()函数；从键盘输入 2，则执行 changebynum()函数；从键盘输入 3，则执行 charu()函数；否则退出该结构。

searchsingle.h 文件中：

- 第 16~41 行，为 switch 结构。若从键盘输入 1，则执行 shuchu()函数；从键盘输入 2，则执行 sort()函数；从键盘输入 3，则执行 sorts()函数；从键盘输入 4，则执行 sortf()函数；从键盘输入 5，则执行 chazhao()函数。

【运行结果】该程序的执行结果如图 18-2 所示。

```
E:\C\Debug\2222.exe
-----欢迎进入学生管理系统-----
*****
1 添加学生资料模块
2 查找学生资料模块
3 修改学生资料模块
4 删除学生资料模块
5 退出
*****
输入要进入的系统模块: _
```

图 18-2 范例 18.1 结果图



18.4 小结

在开发一个系统之前必须先确定该系统的基本架构及其基本功能，再编写相应的代码。例如，在本章中，学生资料包括姓名、年龄、考勤次数、编号、分数等，则将学生定义为结构体类型。学生管理系统包括输入和输出学生资料、添加学生资料、删除学生资料、查询学生资料等功能，则将其编写成不同的函数，再在主函数中调用相应的函数实现不同的功能。

附录 Visual C++操作技巧小代码

本书所有案例都在 Visual C++环境下开发。所以笔者通过这些代码，提供 Visual C++常见操作中的一些小技巧，帮助读者更好的应用 Visual C++。

F1 打开 CD-ROM

```
mciSendString("Set cdAudio door open wait",NULL,0,NULL);
```

F2 关闭 CD_ROM

```
mciSendString("Set cdAudio door closed wait",NULL,0,NULL);
```

F3 关闭计算机

```
OSVERSIONINFO OsVersionInfo; //包含操作系统版本信息的数据结构 OsVersionInfo.dwOSV  
ersionInfoSize = sizeof(OSVERSIONINFO);  
GetVersionEx(&OsVersionInfo); //获取操作系统版本信息  
if(OsVersionInfo.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS)  
{  
    //Windows98, 调用 ExitWindowsEx()函数重新启动计算机  
    DWORD dwReserved;  
    //可以改变第一个参数, 实现注销用户  
    ExitWindowsEx(EWX_REBOOT,dwReserved);  
    //关机、关闭电源等操作  
    //退出前的一些处理程序  
}
```

F4 重启计算机

```
typedef int (CALLBACK *SHUTDOWNDLG)(int); //显示关机对话框函数的指针  
HINSTANCE hInst = LoadLibrary("shell32.dll"); //装入 shell32.dll  
SHUTDOWNDLG ShutDownDialog; //指向 shell32.dll 库显示关机对话框函数的指针  
if(hInst != NULL)  
{  
    //获得函数的地址并调用  
    ShutDownDialog = (SHUTDOWNDLG)GetProcAddress(hInst,(LPSTR)60);  
    (*ShutDownDialog)(0);  
}
```



F5 枚举所有字体

```
LOGFONT lf;
lf.lfCharSet = DEFAULT_CHARSET; //初始化 LOGFONT 结构体
strcpy(lf.lfFaceName, "");
CClientDC dc (this);

//枚举字体
::EnumFontFamiliesEx((HDC) dc, &lf,
(FONTENUMPROC) EnumFontFamProc, (LPARAM) this, 0);
//枚举函数
int CALLBACK EnumFontFamProc(LPENUMLOGFONT lpelf, LPNEWTEXTMETRIC lpntm, DWORD
nFontType, long lparam)
{
    //创建指针
    CDay7Dlg* pWnd = (CDay7Dlg*) lparam;
    //添加字体到列表, m_ctlFontList 是一个列表控件变量
    pWnd->m_ctlFontList.AddString(lpelf->elfLogFont.lfFaceName);
    //返回 1
    return 1;
}
```

F6 一次只运行一个程序实例（如果已运行则退出）

```
if( FindWindow(NULL, "程序标题")) exit(0);
```

F7 得到当前鼠标所在位置

```
CPoint pt;
GetCursorPos(&pt); //得到位置
```

F8 显示和隐藏程序菜单

```
CWnd *pWnd=AfxGetMainWnd();
if(b_m) //隐藏菜单
{
    pWnd->SetMenu(NULL);
    pWnd->DrawMenuBar();
    b_m=false;
}
else
{
    CMenu menu;
    menu.LoadMenu(IDR_MAINFRAME); //显示菜单 也可改变菜单项
    pWnd->SetMenu(&menu);
}
```



```
pWnd->DrawMenuBar();
b_m=true;
menu.Detach();
}
```

F9 获取可执行文件的图标

```
HICON hIcon::ExtractIcon(AfxGetInstanceHandle(),_T("Notepad.exe"),0);
if (hIcon &&hIcon!=(HICON)-1)
{
    pDC->DrawIcon(10,10,hIcon);
}
DestroyIcon(hIcon);
```

F10 窗口自动靠边程序演示

```
BOOL AdjustPos(CRect* lpRect)
{
    //自动靠边
    int iSX=GetSystemMetrics(SM_CXFULLSCREEN);
    int iSY=GetSystemMetrics(SM_CYFULLSCREEN);

    RECT rWorkArea;
    BOOL bResult = SystemParametersInfo(SPI_GETWORKAREA, sizeof(RECT), &rWorkArea, 0);

    CRect rcWA;
    if(!bResult)
    {
        //如果调用不成功就利用 GetSystemMetrics 获取屏幕面积
        rcWA=CRect(0,0,iSX,iSY);
    }
    else
        rcWA=rWorkArea;

    int iX=lpRect->left;
    int iY=lpRect->top;
    if(iX < rcWA.left + DETASTEP && iX!=rcWA.left)
    {
        //调整左
        //pWnd->SetWindowPos(NULL,rcWA.left,iY,0,0,SWP_NOSIZE);
        lpRect->OffsetRect(rcWA.left-iX,0);
        AdjustPos(lpRect);
        return TRUE;
    }
    if(iY < rcWA.top + DETASTEP && iY!=rcWA.top)
    {
```




```
//调整上
//pWnd->SetWindowPos(NULL ,iX,rcWA.top,0,0,SWP_NOSIZE);
lpRect->OffsetRect(0,rcWA.top-iY);
AdjustPos(lpRect);
return TRUE;
}
if(iX+lpRect->Width()>rcWA.right-DETASTEP&& iX!=rcWA.right-lpRect->Width())
{
    //调整右
    //pWnd->SetWindowPos(NULL ,rcWA.right-rcW.Width(),iY,0,0,SWP_NOSIZE);
    lpRect->OffsetRect(rcWA.right-lpRect->right,0);
    AdjustPos(lpRect);
    return TRUE;
}
if(iY+lpRect->Height()>rcWA.bottom-DETASTEP&& iY!=rcWA.bottom-lpRect->Height())
{
    //调整下
    //pWnd->SetWindowPos(NULL ,iX,rcWA.bottom-rcW.Height(),0,0,SWP_NOSIZE);
    lpRect->OffsetRect(0,rcWA.bottom-lpRect->bottom);
    return TRUE;
}
return FALSE;
}
//然后在 ONMOVEING 事件中使用
CRect r=*pRect;
AdjustPos(&r);
*pRect=(RECT)r;
```

F11 删除目录（包含删除子文件夹以及其中的内容）

```
BOOL DeleteDirectory(char *DirName)//如删除 DeleteDirectory("c:\\aaa")
{
    CFileFind tempFind;
    char tempFileFind[MAX_PATH];
    sprintf(tempFileFind,"%s\\*.*",DirName);
    BOOL IsFinded=(BOOL)tempFind.FindFile(tempFileFind);
    while(IsFinded)
    {
        IsFinded=(BOOL)tempFind.FindNextFile();
        if(!tempFind.IsDots())
        {
            char foundFileName[MAX_PATH];
            strcpy(foundFileName,tempFind.GetFileName().GetBuffer(MAX_PATH));
            if(tempFind.IsDirectory())
            {
                char tempDir[MAX_PATH];
```



```
    sprintf(tempDir, "%s\\%s", DirName, foundFileName);
    DeleteDirectory(tempDir);
}
else
{
    char tempFileName[MAX_PATH];
    sprintf(tempFileName, "%s\\%s", DirName, foundFileName);
    DeleteFile(tempFileName);
}
}
}

tempFind.Close();
if(!RemoveDirectory(DirName))
{
    MessageBox(0, "删除目录失败!", "警告信息", MB_OK);
    return FALSE;
}
return TRUE;
}
```

F12 运行其他程序

```
//运行 EMAIL 或网址
char szMailAddress[80];
strcpy(szMailAddress, "mailto:netvc@21cn.com");
ShellExecute(NULL, "open", szMailAddress, NULL, NULL, SW_SHOWNORMAL);

//运行可执行程序
WinExec("notepad.exe", SW_SHOW);
```

F13 动态增加或删除菜单

(1) 增加菜单

```
//添加
CMenu *mainmenu;
mainmenu=AfxGetMainWnd()->GetMenu(); //得到主菜单
(mainmenu->GetSubMenu(0))->AppendMenu(MF_SEPARATOR); //添加分隔符
(mainmenu->GetSubMenu(0))->AppendMenu(MF_STRING, ID_APP_ABOUT, _T("Always on &Top"));
//添加

DrawMenuBar(); //重画菜单
```

(2) 删除菜单

```
//删除
```



```
CMenu *mainmenu;
mainmenu=AfxGetMainWnd()->GetMenu();           //得到主菜单
CString str ;
for(int i=(mainmenu->GetSubMenu (0))->GetMenuItemCount()-1;i>=0;i--)
    //取得菜单的项数
{
    (mainmenu->GetSubMenu (0))->GetMenuString(i,str,MF_BYPOSITION);
    //将指定菜单项的标签复制到指定的缓冲区。MF_BYPOSITION 的解释见上
    if(str=="Always on &Top")                    //如果是刚才增加的菜单项，则删除
    {
        (mainmenu->GetSubMenu (0))->DeleteMenu(i,MF_BYPOSITION);
        break;
    }
}
```

F14 测试 ALT 键是否按下

```
GetKeyState(VK_MENU);
GetAlt();
```

F15 检查是否按下鼠标左键

```
if((nFlags&MK_LBUTTON)==MK_LBUTTON)
```

F16 检查键盘输入

在 OnKeyDown 中的参数 nChar 是一个数值，当显示的时候，需要转换成字符，使用如下的命令：

```
char lsChar;
lsChar=char(nChar);
if(lsChar=='A');
{
...
}
```

F17 调用另一个函数::GetKeyState()，用一个特定的键代码来确定键是否被按下

如果::GetKeyState 函数的返回值是负的，表示该键被按下。如果返回值是非负的，表示该键未被按下。如果要确定 Shift 键是否被按下，可以使用下面的代码：

```
if(::GetKeyState(VK_SHIFT) <0)
{
    AfxMessageBox("shift is pressed");
}
```



F18 如何在编程的过程中随时结束应用程序（常规）

（1）需要向窗口发送 WM_CLOSE/WM_QUIT 消息，调用 CWnd::OnClose 成员函数并允许对用户提示是否保存修改过的数据。

```
AfxGetMainWnd()->SendMessage(WM_CLOSE); //别忘了先得到当前窗口的指针
```

（2）使用函数：void PostQuitMessage(int nExitCode);。

（3）使用标准函数：void exit(int status);。

F19 十六进制转化成十进制小数的问题

浮点技术法，如：

```
DWORD dw=0x3CF5C28F;  
float d=*(float*)&dw;//0.03;
```

F20 在桌面上任意位置写字

```
HDC deskdc = ::GetDC(0);  
CString stext = "我的桌面";  
::TextOut(deskdc,100,200,stext,stext.GetLength());  
::ReleaseDC(0,deskdc);
```

F21 隐藏桌面图标并禁用右键功能菜单

```
HWND Hwd = ::FindWindow("Progman", NULL);  
if (bShown)  
::ShowWindow(Hwd, SW_HIDE);  
else  
::ShowWindow(Hwd, SW_SHOW);  
  
bShown = !bShown;
```

F22 删除非空文件夹

```
SHFILEOPSTRUCT shfileop;  
shfileop.hwnd = NULL;  
shfileop.wFunc = FO_DELETE ;  
shfileop.fFlags = FOF_SILENT | FOF_NOCONFIRMATION;  
shfileop.pFrom = "c:\\temp"; //要删除的文件夹  
shfileop.pTo = "";  
shfileop.lpszProgressTitle = "";  
shfileop.fAnyOperationsAborted = TRUE;  
int nOK = SHFileOperation(&shfileop);
```



F23 快速从数字的字符串中提取出特定长度的数字

```
int a[4];
//按指定长度分隔
sscanf("2004115819185", "%07d%02d%02d%02d", &a[0], &a[1], &a[2], &a[3]);
```

或:

```
CString s="aaa,bbb,ccc,ddd";
char a1[4],a2[4],a3[4],a4[4]; //这里要注意多留点空间以存放各子串的长度
sscanf(s,"%[^,],[^,],[^,],[^,]",a1,a2,a3,a4); //按指定字符（这里是逗号）分隔
AfxMessageBox(a4); //显示 ddd
```

F24 根据 ComboBox 加入的字符串长度自动调整 ComboBox 控件宽度

```
//这里假设为 ComboBox 加入两个字符串
CString str1="中华人民共和国中华人民共和国",str2="1234567890123 中国 89012345678";
//m_combo 为绑定在组合框控件的变量
m_combo.AddString(str1);
m_combo.AddString(str2);
//根据加入的字符串长度（以字节为单位）和组合框使用的默认字体
int len=str1.GetLength()*6.2;
m_combo.SetDroppedWidth(len);
```

F25 快速从得到的路径文件名中分离出盘符、路径名、文件名和后缀名

```
char path_buffer[_MAX_PATH];
char drive[_MAX_DRIVE];
char dir[_MAX_DIR];
char fname[_MAX_FNAME];
char ext[_MAX_EXT];
GetModuleFileName(0,path_buffer,_MAX_PATH);
_splitpath( path_buffer, drive, dir,fname , ext); //用这个函数转换
```

F26 获取硬盘剩余空间

```
/*
driver 为空获取 conf.ini DISK 指定的磁盘剩余空间和 (C:D:F:)
driver 不为空获取此硬盘剩余空间
*/
ULONG ReturnFreeSpace(LPSTR driver/* =NULL */)
{
    ULONG total=0;
```



```
TCHAR DiskStr[3];
TCHAR szDrives[128];
PULARGE_INTEGER pDisknum;
pDisknum= new ULARGE_INTEGER;
TCHAR* pDrive;
GetPrivateProfileString(LocalIP,"DISK","",szDrives,128,".\\conf.ini");
pDrive = szDrives;
CString tempStr;
if (NULL==driver)           //判断磁盘
{
    while (*pDrive)         //循环判断并合计
    {
        DiskStr[0]=*pDrive;
        DiskStr[1]=': ';
        DiskStr[2]='\0';
        if (GetDiskFreeSpaceEx(DiskStr,pDisknum,NULL,NULL))
        {
            total+=(pDisknum->QuadPart/(1024)); //K bytes
        }
        pDrive+=2;
    }
}
else
{
    GetDiskFreeSpaceEx(driver,pDisknum,NULL,NULL);
    total=pDisknum->QuadPart/(1024*1024); //M bytes
}
MEMORY_DELETE(pDisknum);
return total;               //返回磁盘空间总数
}
```